Comment simuler une évacuation d'un lieu clos : étude de flux de personnes.

LAGNEAU Clément

Sommaire

- Introduction
- Simulation en python
- Critères de satisfaction
- Algorithme génétique
- Impact humain
- Conclusion

Introduction

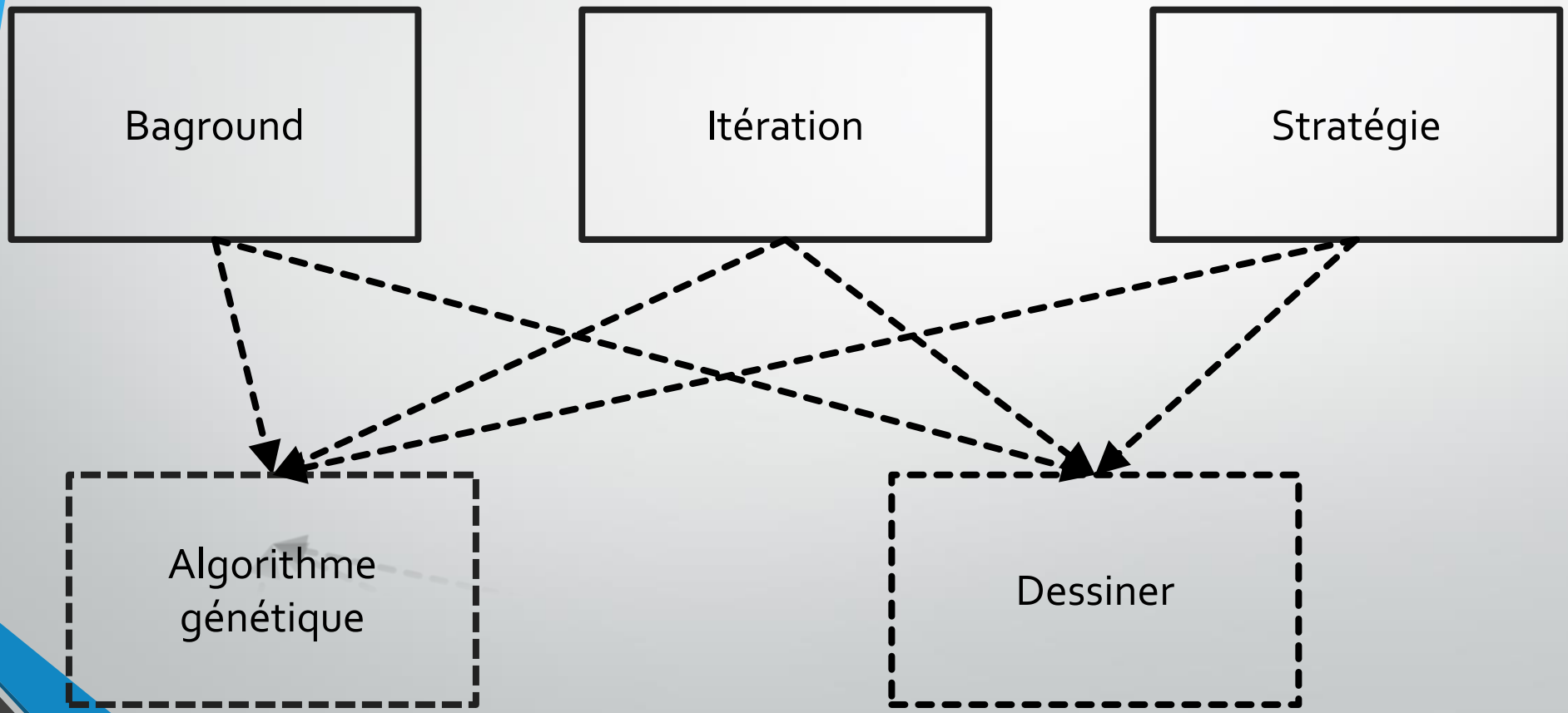
Simulation d'évacuation

Algorithme génétique

Lieu donné ⟶ Stratégie ⟶ Stratégie adaptée


Pathfinder

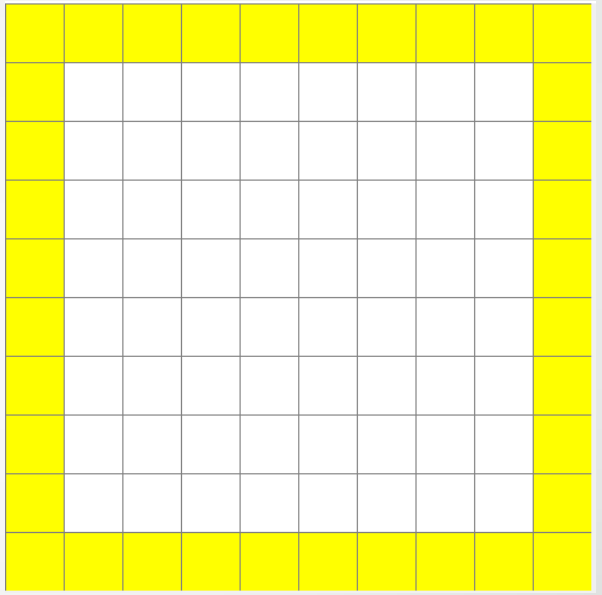
Implémentation d'une simulation d'évacuation en Python



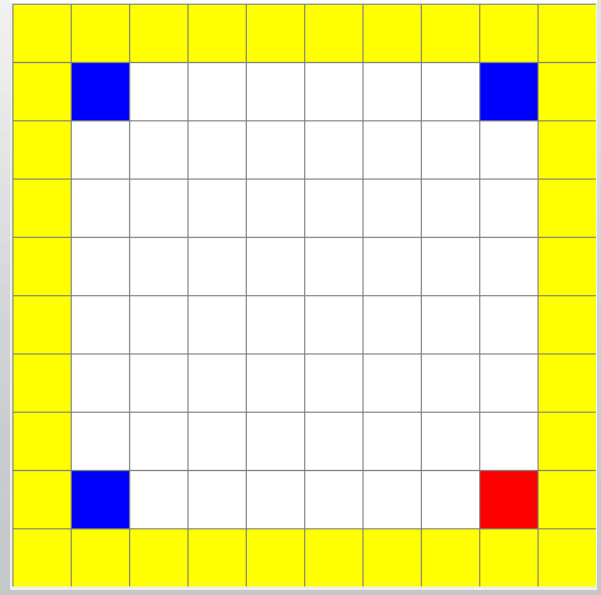
Baground

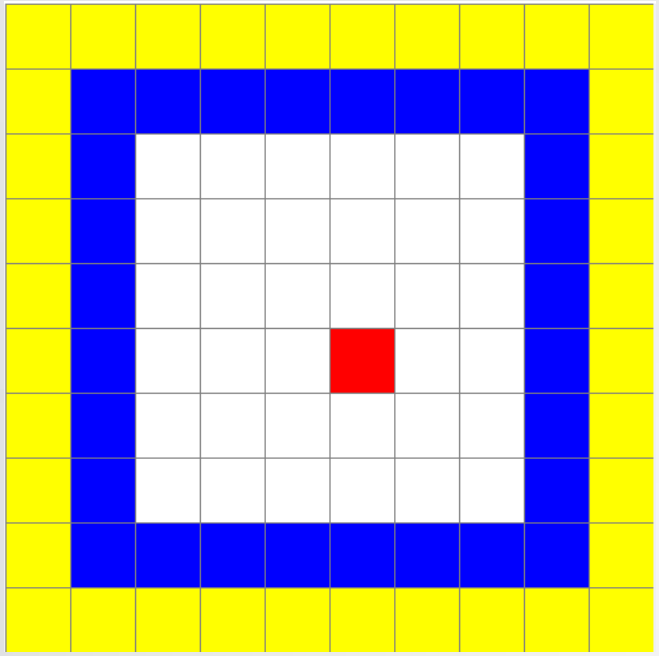
- Créer les fonds

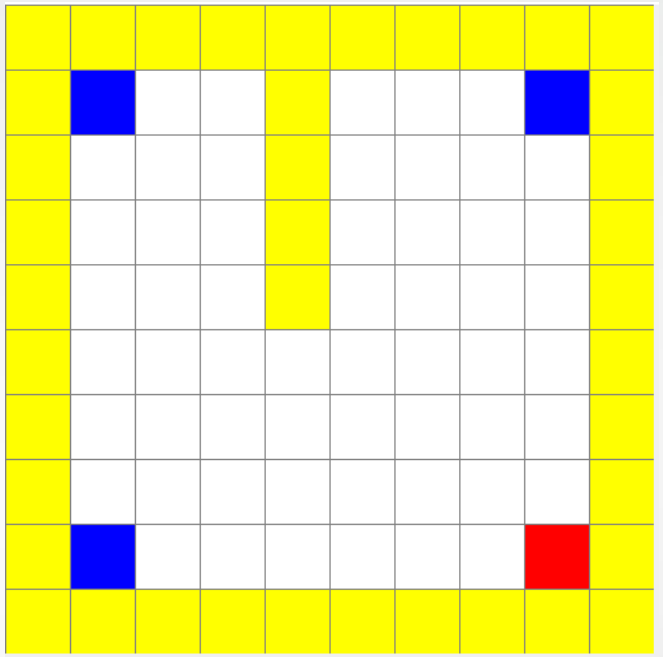
```python
def baground(n):
    """
    Cree un baground avec des murs sur les cotes
    """
    t=[[0 for x in range(n)] for x in range(n)]
    for x in range(n):
        t[0][x]=2
        t[x][0]=2
        t[n-1][x]=2
        t[x][n-1]=2
    return(t)
```

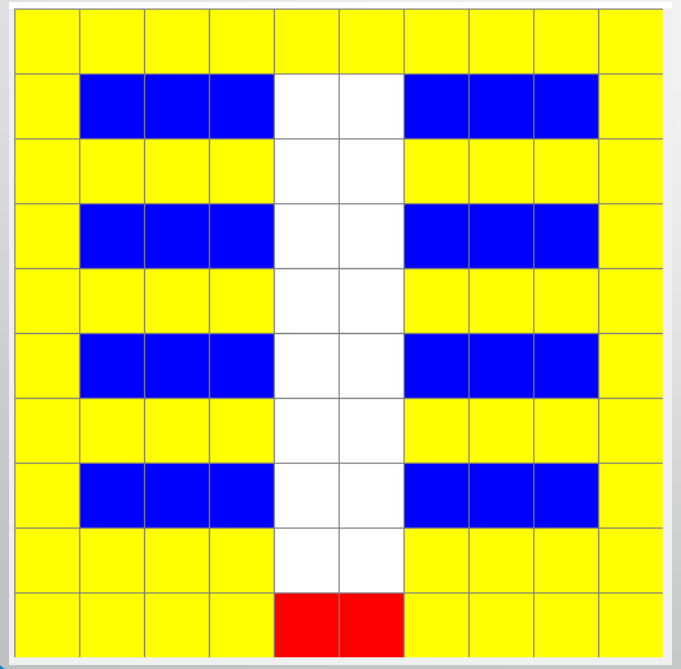


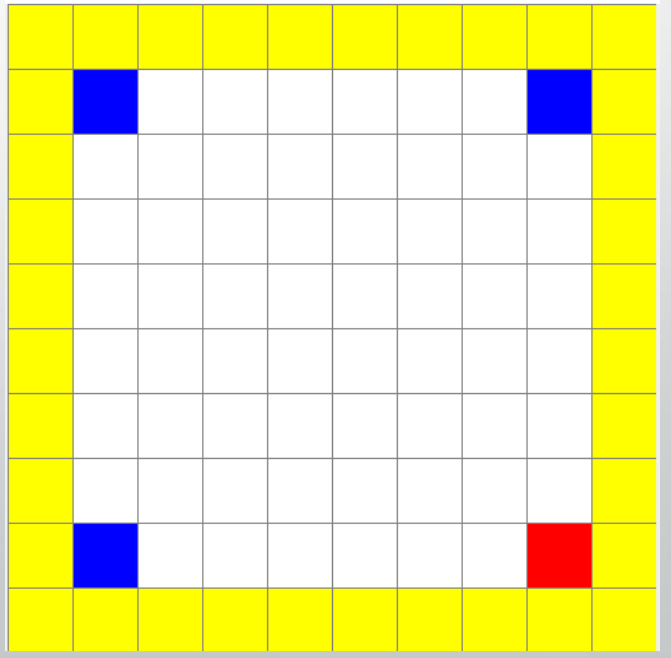
```python
def baground_2(n):
    """
    Cree un baground avec des 1 dans les coins et une sortie dans le dernier coin
    """
    a=baground(n)
    a[n-2][n-2]=-1
    a[1][1]=1
    a[1][n-2]=1
    a[n-2][1]=1
    return(a)
```







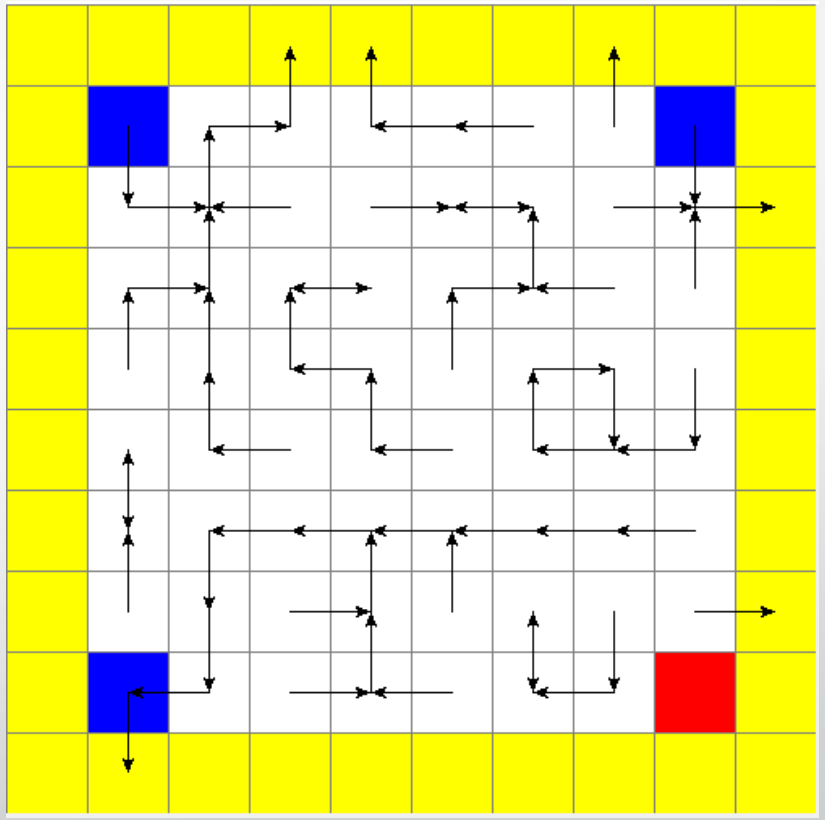




Strategie

```python
def strategie(t):
    """
    Renvoie une strategie adaptee au baground t
    """
    n=len(t)
    strat=[["" for x in range(n)] for x in range(n)]
    for x in range(n):
        for y in range(n):
            if t[x][y] == 2:
                strat[x][y]="i"
            elif t[x][y] == -1:
                strat[x][y]="s"
            else:
                r=random.random()
                if r<=0.25:
                    strat[x][y]="h"
                elif 0.25<r<=0.5:
                    strat[x][y]="b"
                elif 0.5<r<=0.75:
                    strat[x][y]="d"
                else:
                    strat[x][y]="g"
    return(strat)
```

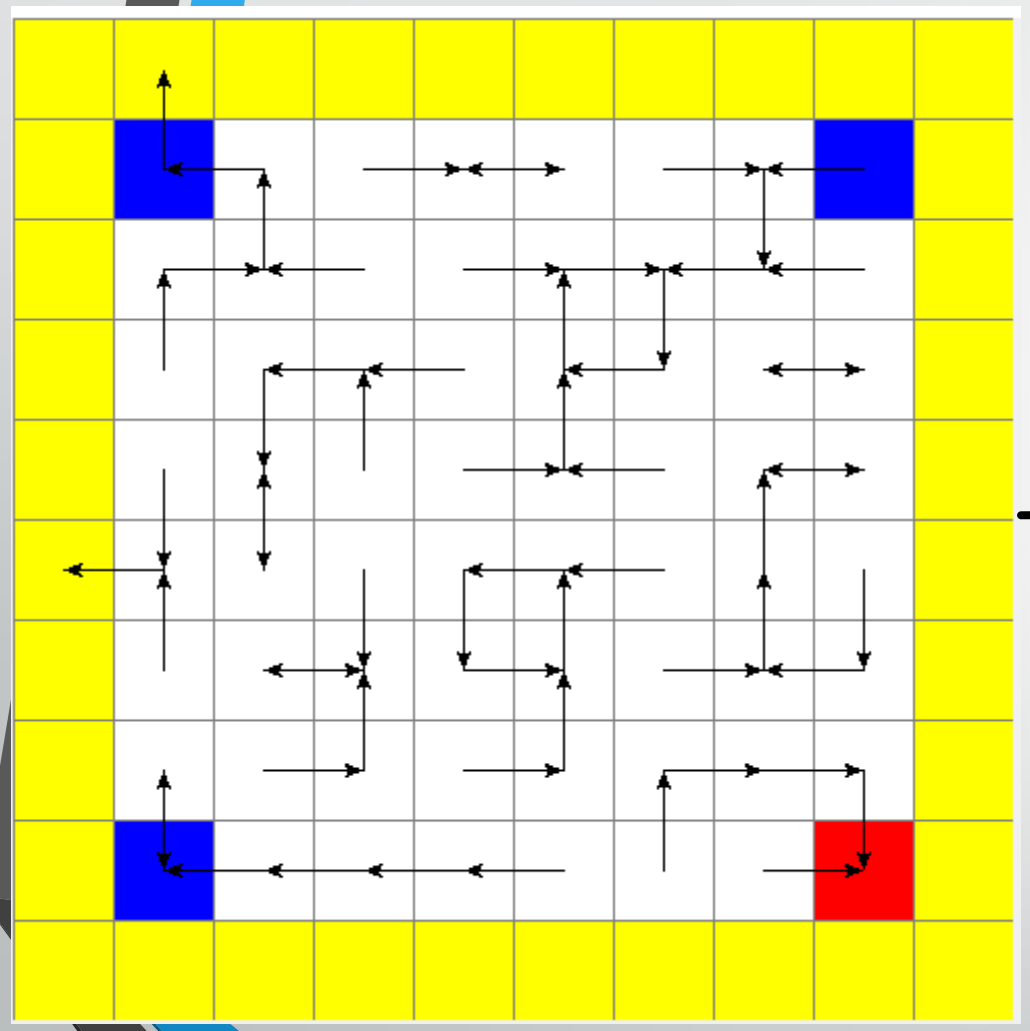
```python
def strategie_case(t,x,y):
    """
    Renvoie une strategie adaptee a la case x y du baground t
    """
    if t[x][y] == "i":
        return("i")
    elif t[x][y] == "s":
        return("s")
    else:
        r=random.random()
        if r<=0.25:
            return("h")
        elif 0.25<r<=0.5:
            return("b")
        elif 0.5<r<=0.75:
            return("d")
        else:
            return("g")
```



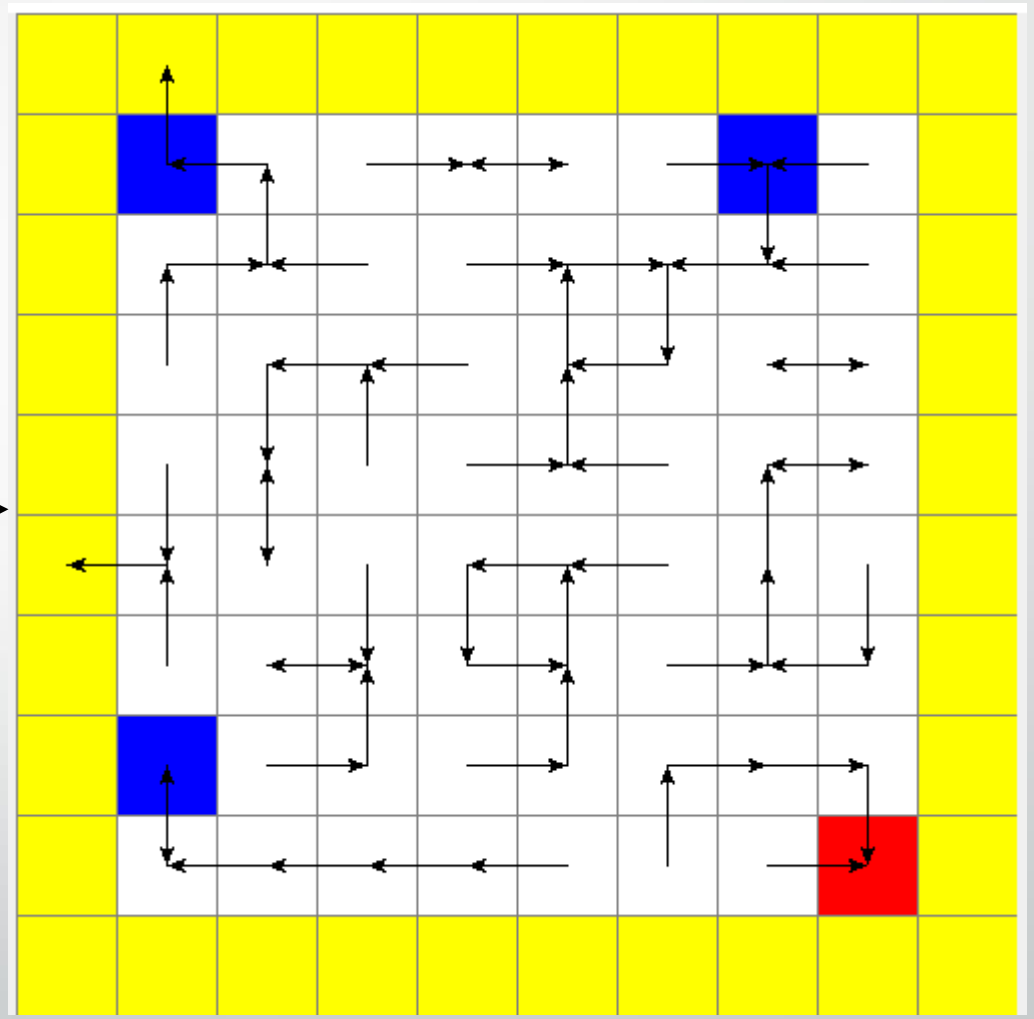
```python
from copy import deepcopy

def avance(a,b,score):
    """
    Renvoie le tableau et le score avance avec une iteration
    """
    n=len(a)
    c=deepcopy(a)
    for ligne in range(n):
        for colonne in range(n):
            if a[ligne][colonne]==1:
                # On avance
                if b[ligne][colonne]=="d":
                    if a[ligne][colonne+1]==0 and c[ligne][colonne+1]==0:
                        c[ligne][colonne]=0
                        c[ligne][colonne+1]=1
                    if a[ligne][colonne+1]==-1:
                        c[ligne][colonne]=0
                        score+=1
                if b[ligne][colonne]=="g":
                    if a[ligne][colonne-1]==0 and c[ligne][colonne-1]==0:
                        c[ligne][colonne]=0
                        c[ligne][colonne-1]=1
                    if a[ligne][colonne-1]==-1:
                        c[ligne][colonne]=0
                        score+=1
                if b[ligne][colonne]=="b":
                    if a[ligne+1][colonne]==0 and c[ligne+1][colonne]==0:
                        c[ligne][colonne]=0
                        c[ligne+1][colonne]=1
                    if a[ligne+1][colonne]==-1:
                        c[ligne][colonne]=0
                        score+=1
                if b[ligne][colonne]=="h":
                    if a[ligne-1][colonne]==0 and c[ligne-1][colonne]==0:
                        c[ligne][colonne]=0
                        c[ligne-1][colonne]=1
                    if a[ligne-1][colonne]==-1:
                        c[ligne][colonne]=0
                        score+=1
    return(c,score)

```

Iteration

- Permet d’avancer d’une iteration



Avance



Dessiner

- Bibliothèque Tkinter

```
import tkinter
import iteration
from copy import deepcopy
import baground
import strategie

k=50

#a=baground.baground_test1(n)
#b=strategie.strategie(a)

dico = {-1 : "red", 2 : "yellow",1 :"blue", 0: "white"}

score = 0

def init():
    for x in range(n):
        for y in range(n):
            couleur[y][x]=canvas.create_rectangle((x*k, y*k,(x+1)*k, (y+1)*k), outline="gray", fill="red")

def calcul():
    global score
    global a
    c,score=iteration.avance(a,b,score)
    for x in range(n):
        for y in range(n):
            canvas.itemconfig(couleur[x][y], fill=dico[c[x][y]])
    a=deepcopy(c)

def final():
    calcul()
    fenetre.after(1000, final)

couleur = [[0 for x in range(n)] for y in range(n)]

fenetre = tkinter.Tk()
canvas = tkinter.Canvas(fenetre, width=k*n, height=k*n, highlightthickness=0)
canvas.pack()
init()

for x in range(n):
    for y in range(n):
        if b[x][y] == "h":
            canvas.create_line(((y+0.5)*k, (x+0.5)*k, (y+0.5)*k, (x-0.5)*k),arrow='last')
        if b[x][y] == "b":
            canvas.create_line(((y+0.5)*k, (x+0.5)*k, (y+0.5)*k, (x+1.5)*k),arrow='last')
        if b[x][y] == "d":
            canvas.create_line(((y+0.5)*k, (x+0.5)*k, (y+1.5)*k, (x+0.5)*k),arrow='last')
        if b[x][y] == "g":
            canvas.create_line(((y+0.5)*k, (x+0.5)*k, (y-0.5)*k, (x+0.5)*k),arrow='last')

final()
fenetre.mainloop()
```

Algorithme génétique

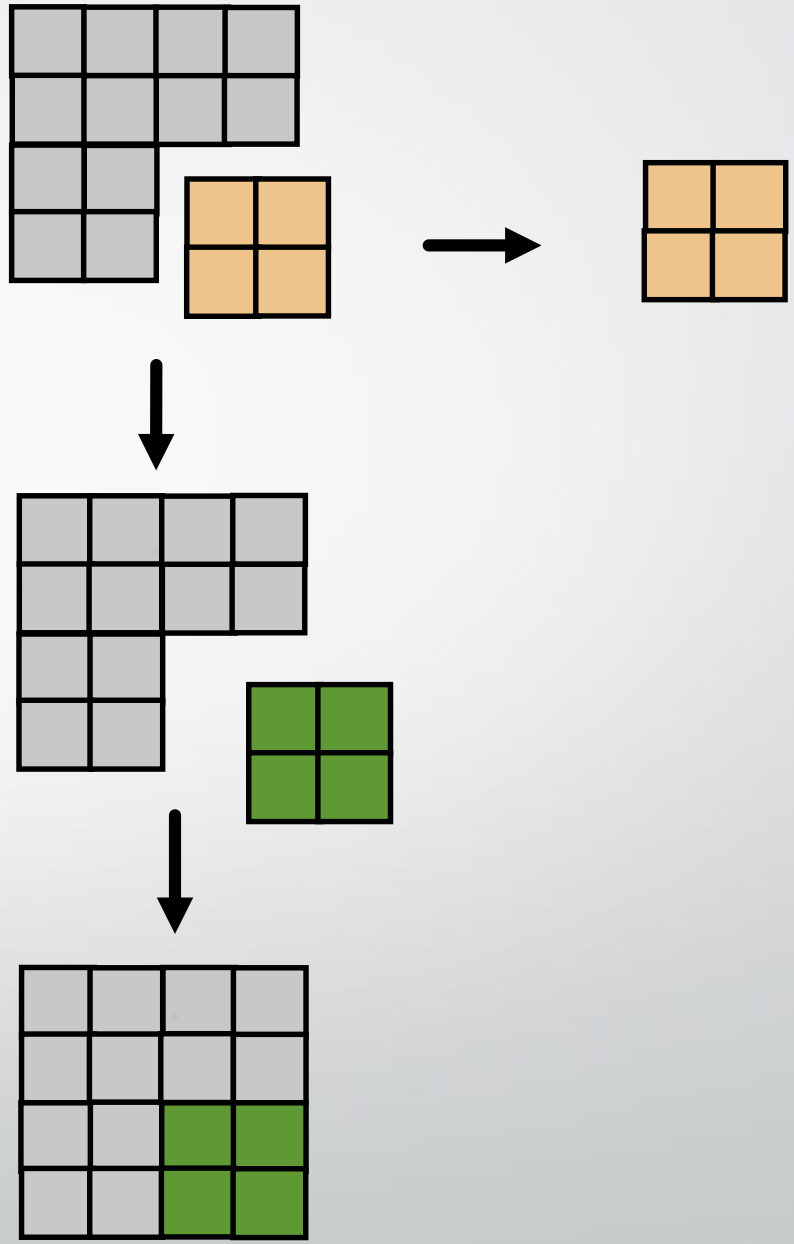
Initialisation

Population

Classement

Mutations

Résultat

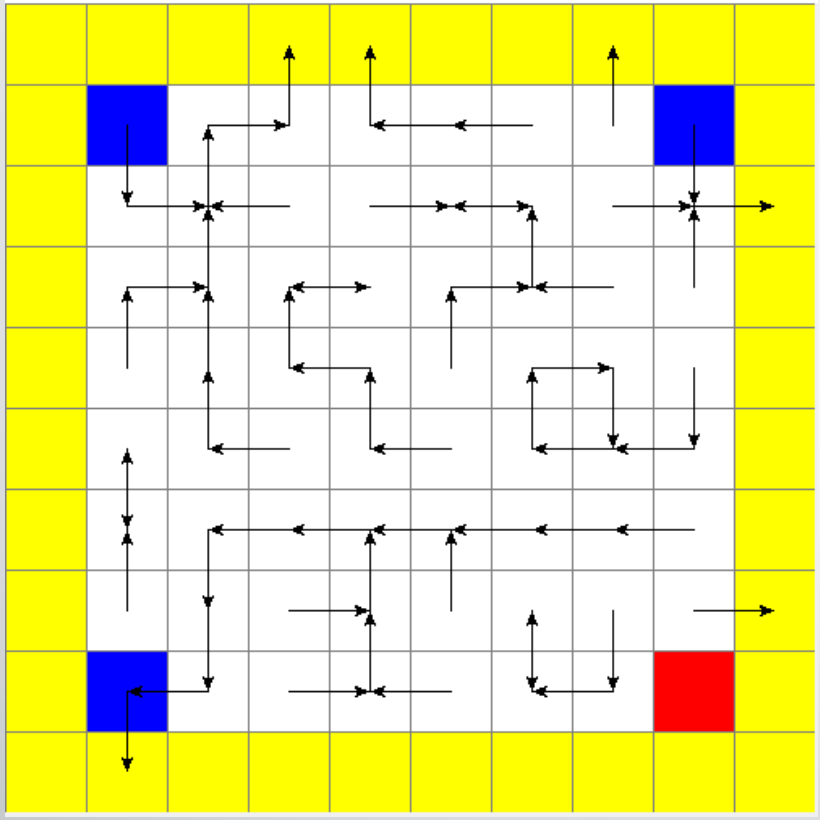


Critère de satisfaction

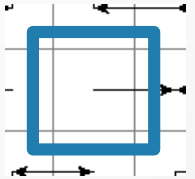
- Le nombre de personne ayant atteint la sortie
- Le nombre d'itérations avant que toutes les personnes soient sorti

```python

#Nombre
n= 10

#Rapport visible/n
k = 50

#taille de liste
i=30

it=n*n*2

import os
import random
import baground
import iteration
import strategie
from copy import deepcopy

def cle(x):
    return(x[0:2])

def nbvivant(t,n):
    res=0
    for i in range(n):
        for j in range(n):
            if t[i][j]==1:
                res += 1
    return(res)

bag=baground.baground_3

vivants = nbvivant(bag(n),n)

# nb de changement dynamique
changement = n

```

```python
def principal(nbiteration,a,b):
    """
    Renvoie le nombre max d iteration et le score pour une table
    """
    c = deepcopy(a)
    score=0
    i=0
    while i<nbiteration and score<vivants:
        c,score=iteration.avance(c,b,score)
        i+=1
    return(i,score)

```



Individu de la population



Gène

Initialisation

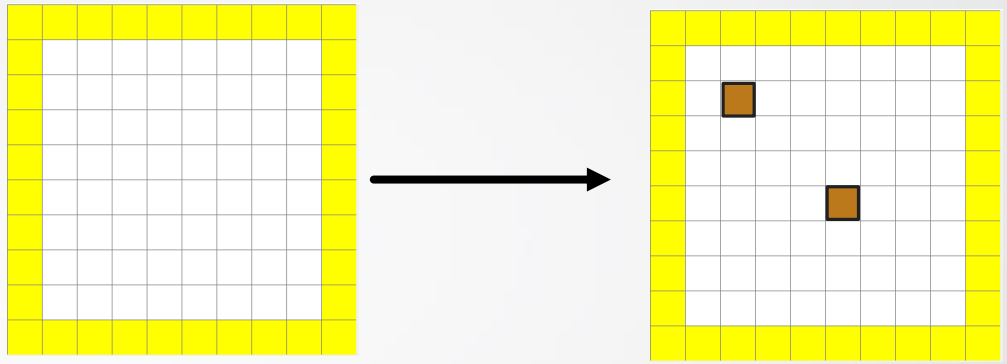
```
45 def meilleur_init():
46     """
47     Renvoie la liste initiale avec comme liste de sortie :
48         [0]: score
49         [1]: nbiteration max
50         [2]: la table de strategie
51     """
52     liste = []
53     for x in range(i):
54         a = bag(n)
55         b = strategie.strategie(a)
56         c = principal(it,a,b)
57         liste.append((c[1],c[0],b))
58     liste=sorted(liste,key=cle,reverse=True)
59     return(liste)
60
```

0	tuple	3	(2, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'b', 'h', 'd ...
1	tuple	3	(1, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'd', 'h', 'h ...
2	tuple	3	(0, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'b', 'd', 'g ...
3	tuple	3	(0, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'g', 'g', 'b ...
4	tuple	3	(0, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'h', 'b', 'b ...
5	tuple	3	(0, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'g', 'b', 'd ...
6	tuple	3	(0, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'g', 'd', 'b ...
7	tuple	3	(0, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'h', 'd', 'd ...
8	tuple	3	(0, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'd', 'h', 'b ...
9	tuple	3	(0, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'g', 'g', 'b ...
10	tuple	3	(0, 100, [['i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i', 'i'], ['i', 'h', 'g', 'b ...

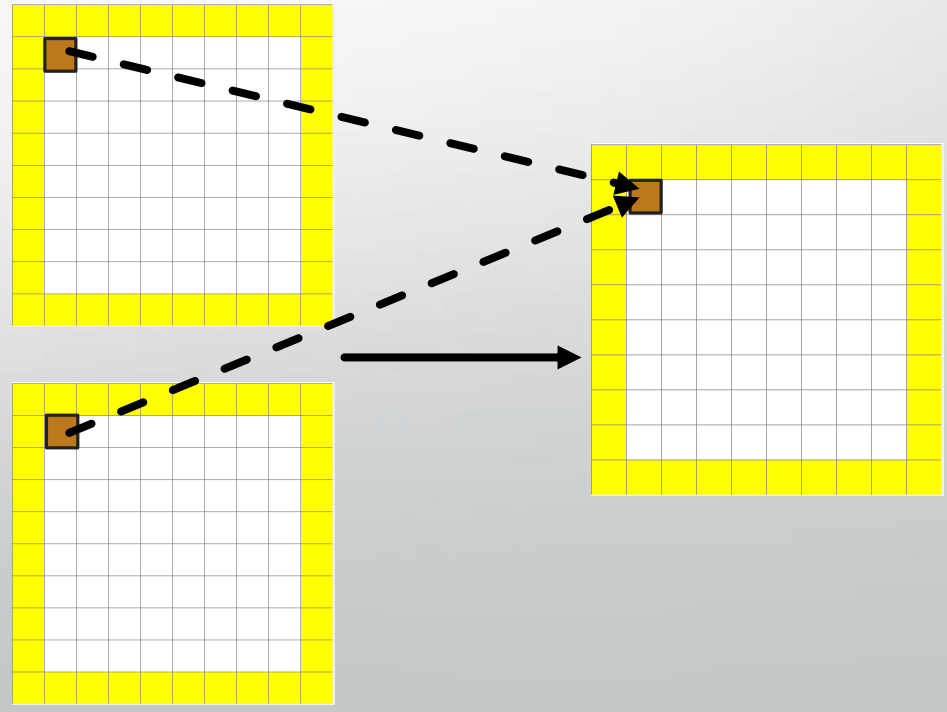
Mutations

```
60
61 def fusion_liste(liste,n):
62     """
63     Renvoie la liste apres la mutation
64     """
65     l=[]
66     for x in range(3*i//4):
67         l.append(deepcopy(liste[x][2]))
68     choix=random.sample(liste,i//8)
69     for x in range(i//8):
70         l.append(fusion(choix[x][2],n))
71     for x in range(i//8):
72         l.append(fusion2(liste[x][2],liste[i//8+x][2],n))
73     return(l)
74
```

```python
def fusion(liste,n):
    """
    Renvoie la liste en effectuant /changement/ changements sur les cases
    """
    r=deepcopy(liste)
    for k in range(changement):
        x=random.randint(0,n-1)
        y=random.randint(0,n-1)
        r[x][y]=strategie.strategie_case(liste,x,y)
    return(r)
```



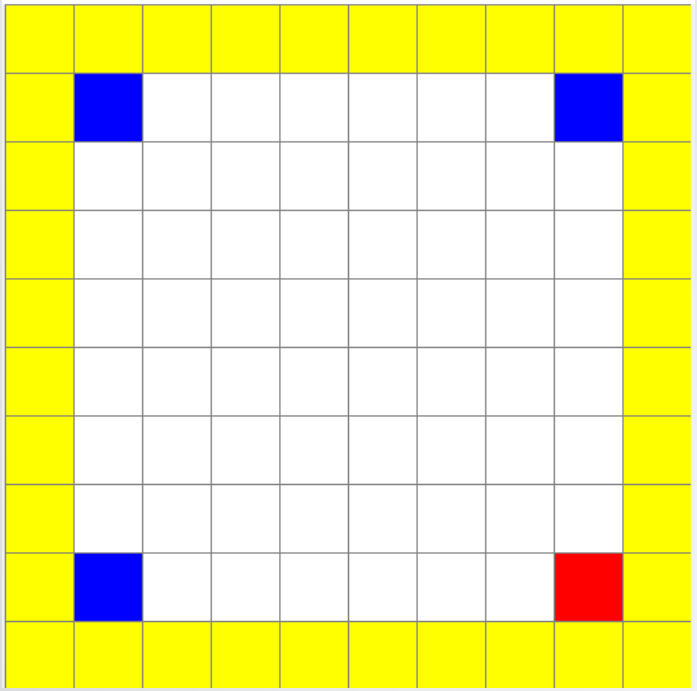
```python
def fusion2(liste1,liste2,n):
    """
    Renvoie la liste fusionne de l1 et l2 suivant une proba 1/2
    """
    r=deepcopy(liste1)
    for x in range(10):
        for y in range(10):
            t=random.random()
            if t<0.5:
                r[x][y]=liste2[x][y]
    return(r)
```

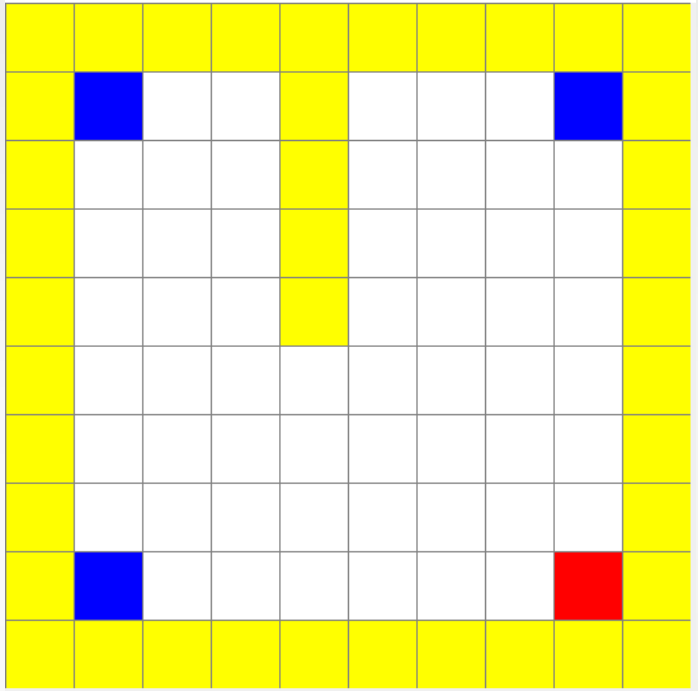


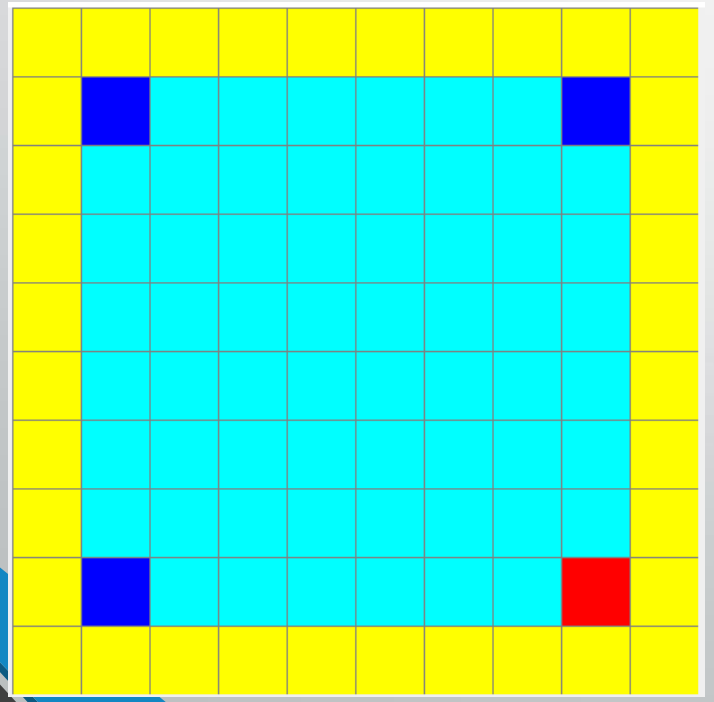
Classement

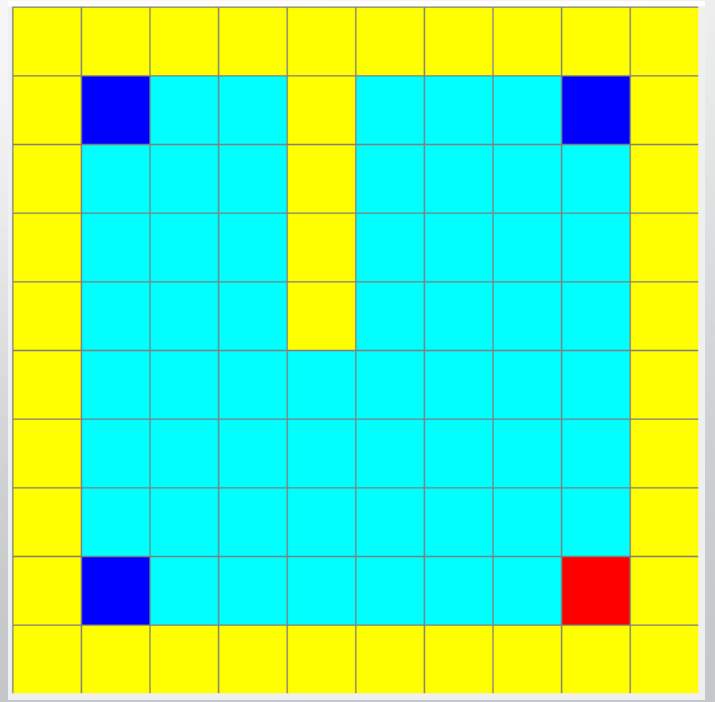
```
75 def meilleur_apres(liste):
76     """
77     Renvoie la liste apres une iteration de l algorithme genetique
78     """
79     liste_apres=[]
80     for b in liste:
81         a = bag(n)
82         c = principal(it,a,b)
83         liste_apres.append((c[1],c[0],b))
84     liste_apres=sorted(liste_apres,key=cle,reverse=True)
85     return(liste_apres)
86
```

```
124 en_cours=[]
125 lapres=meilleur_init()
126 indice=0
127
128 nom="Res_"+str(n)+"_baground_1_"+str(i)+"_1"
129
130
131 os.chdir("D:/Cours/MPBIS/TIPE/")
132 res = open(nom,"w")
133 res.close()
134
135 while lapres[0][0]!=vivants and indice<50000:
136     res = open(nom,"a")
137     res.writelines(str((indice,lapres[0][0],changement))+"\n")
138     res.close()
139     changement = vivants-lapres[0][0]
140     print(indice,lapres[0][0],changement)
141     en_cours=fusion_liste(lapres,n)
142     lapres=meilleur_apres(en_cours)
143     indice +=1
144
145 res = open(nom,"a")
146 res.writelines(str(lapres[0][2]))
147 res.close()
148
149
150 print(lapres[0][2])
151
```

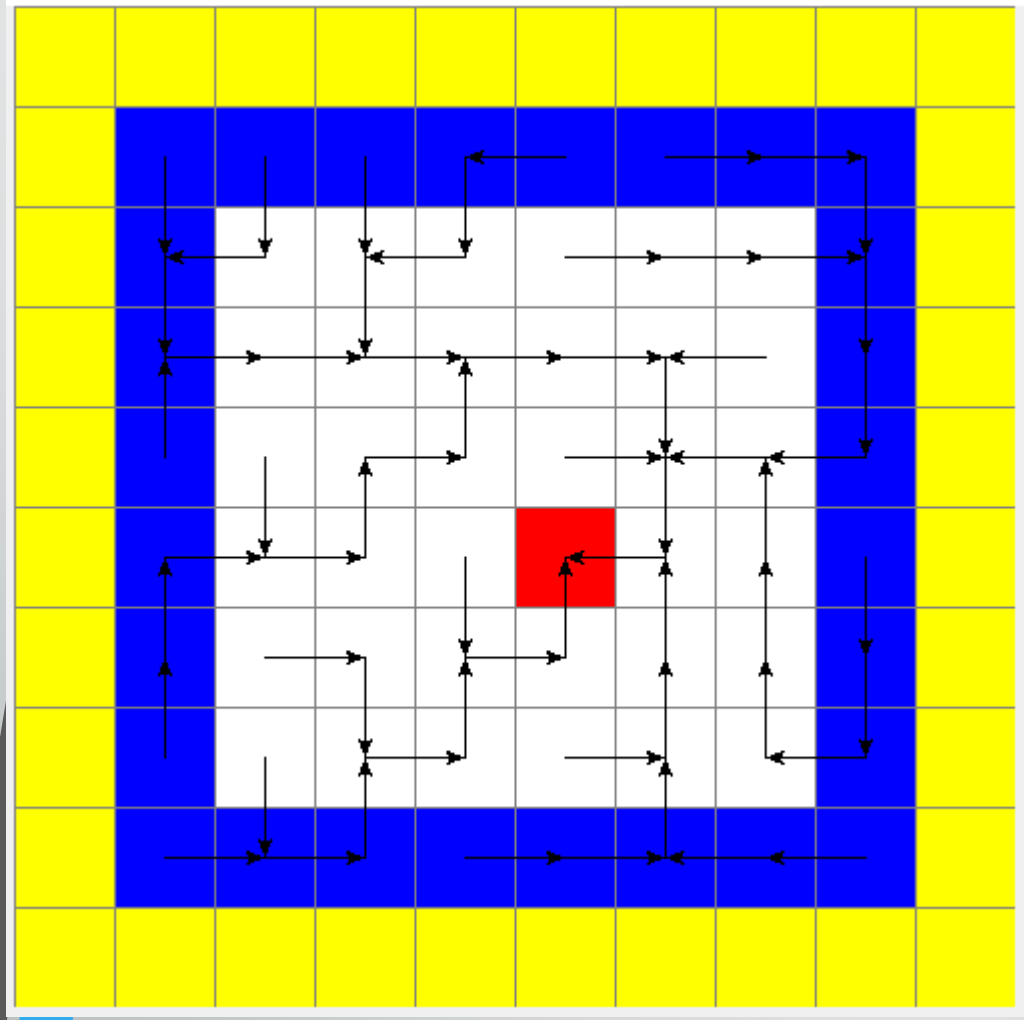




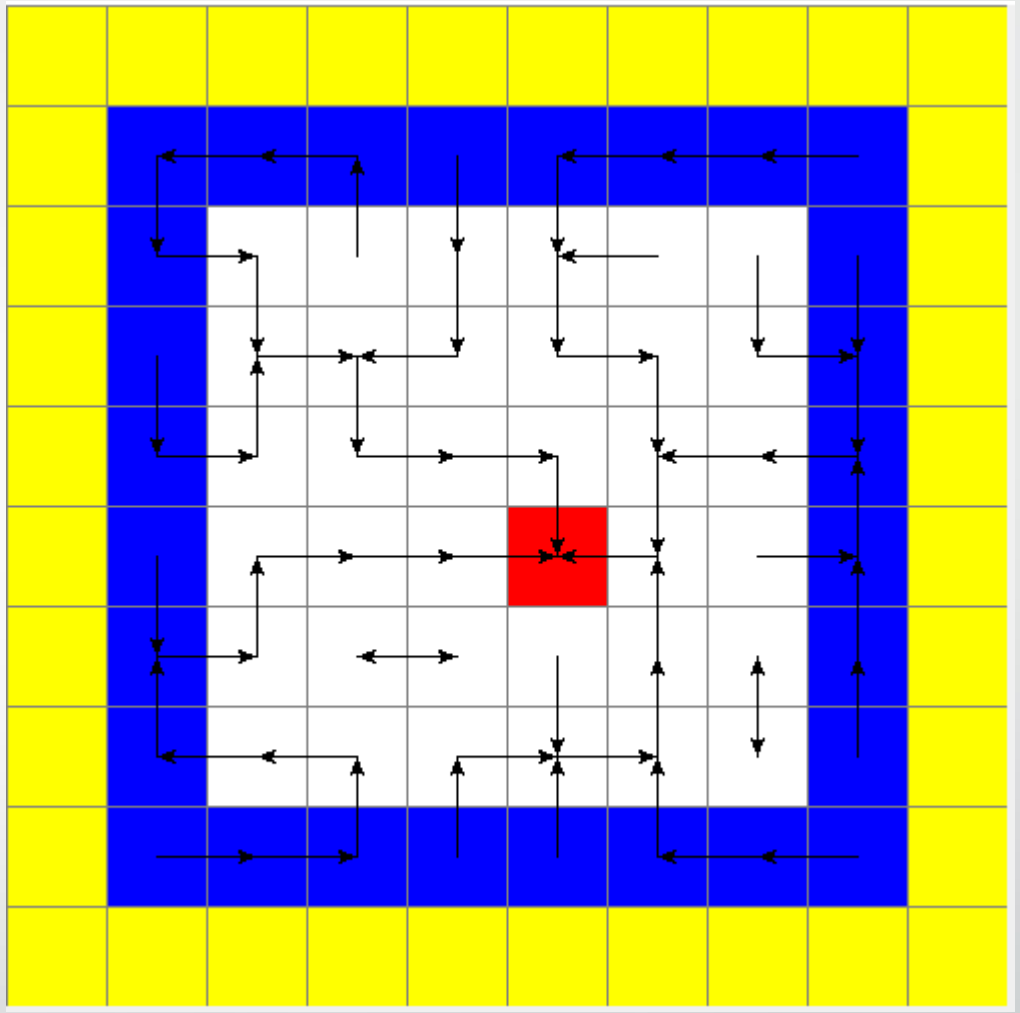




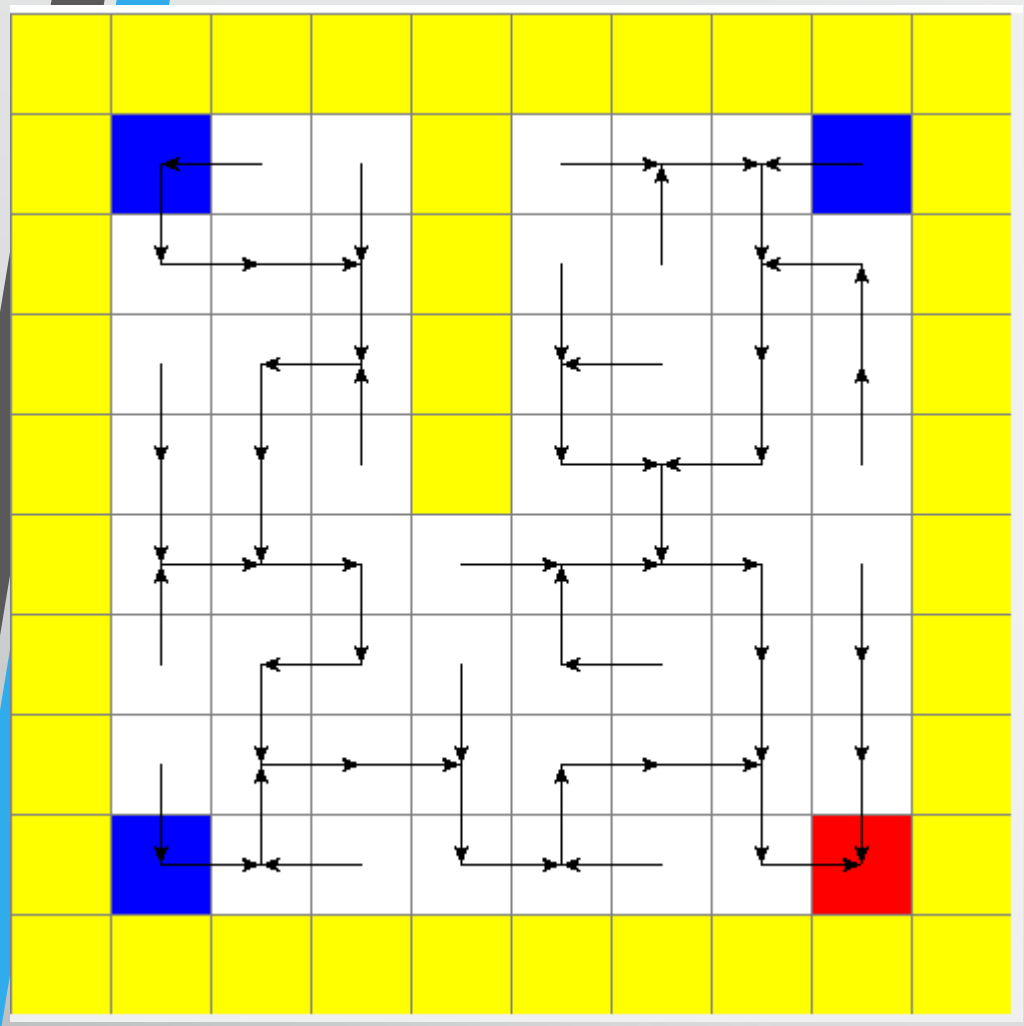
Resultats

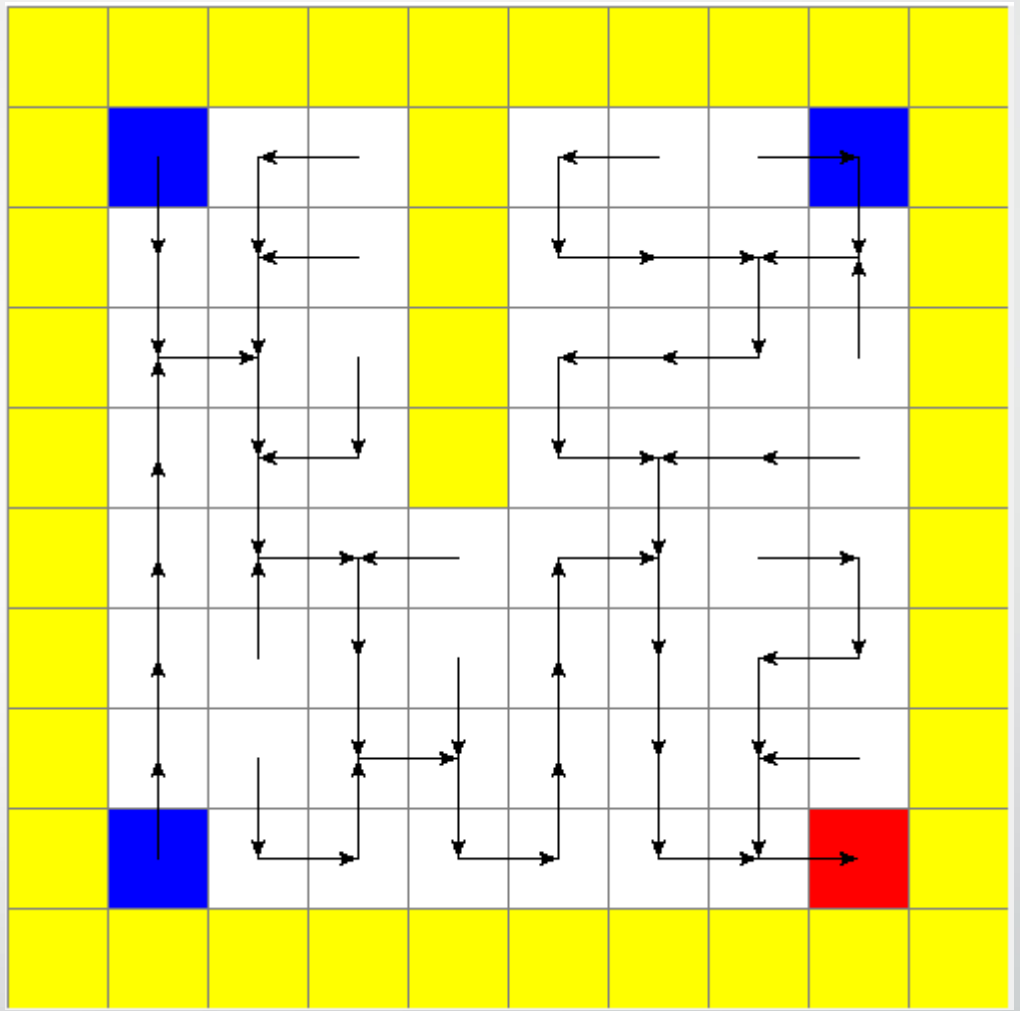


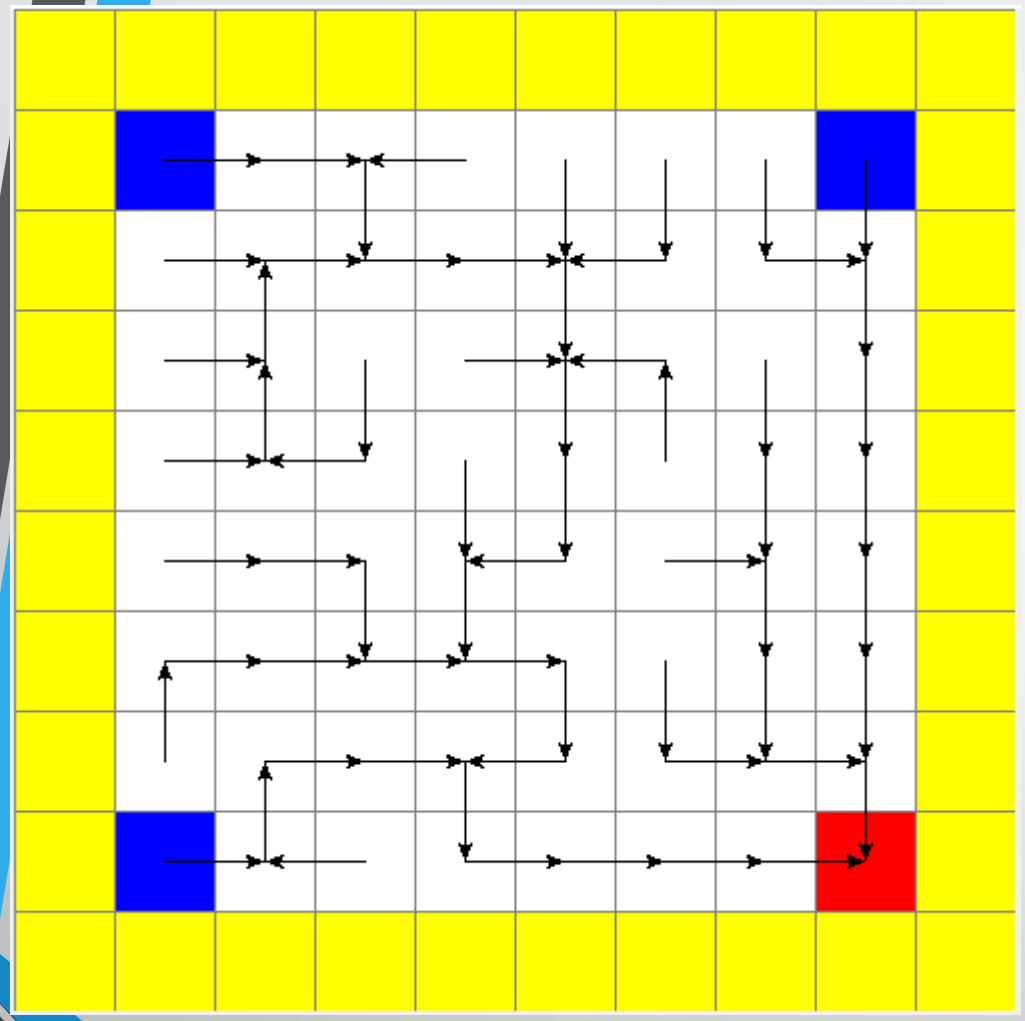
Avec 1 - 90 iterations

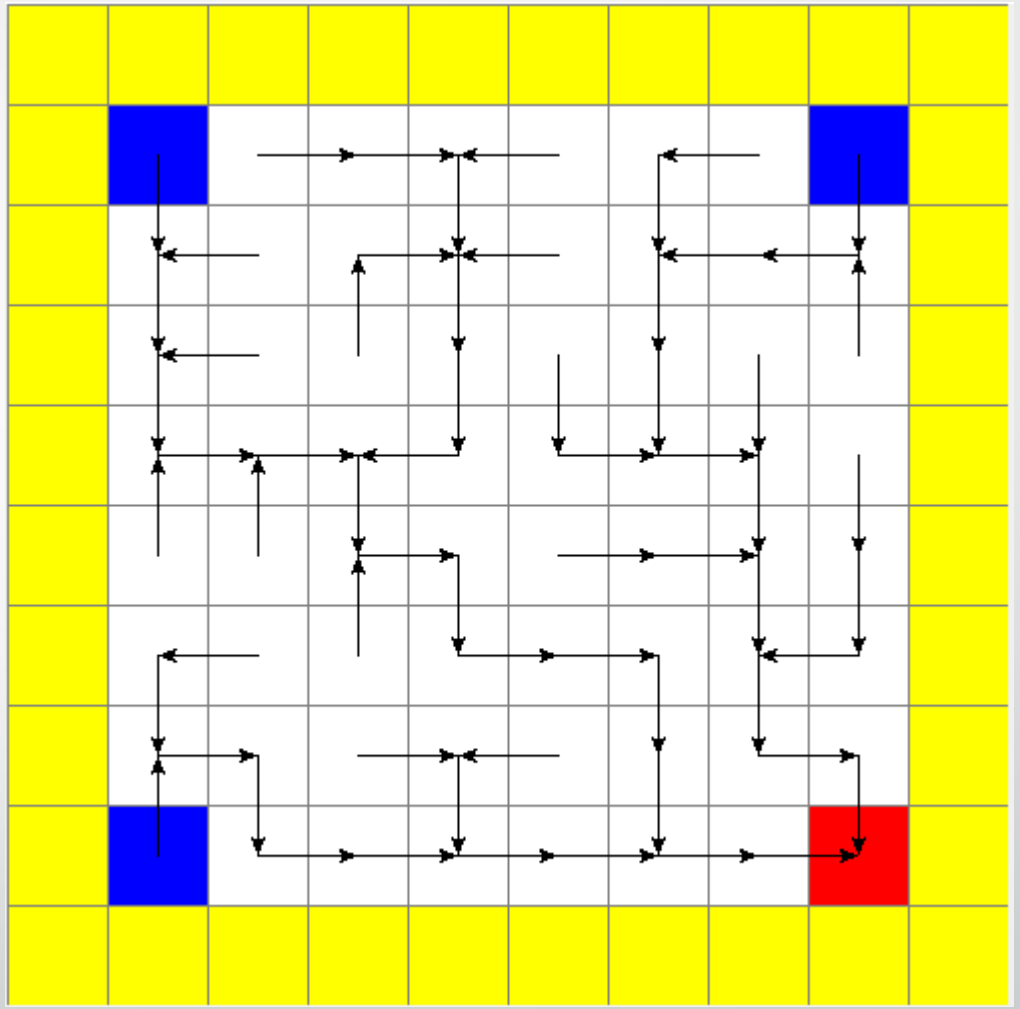


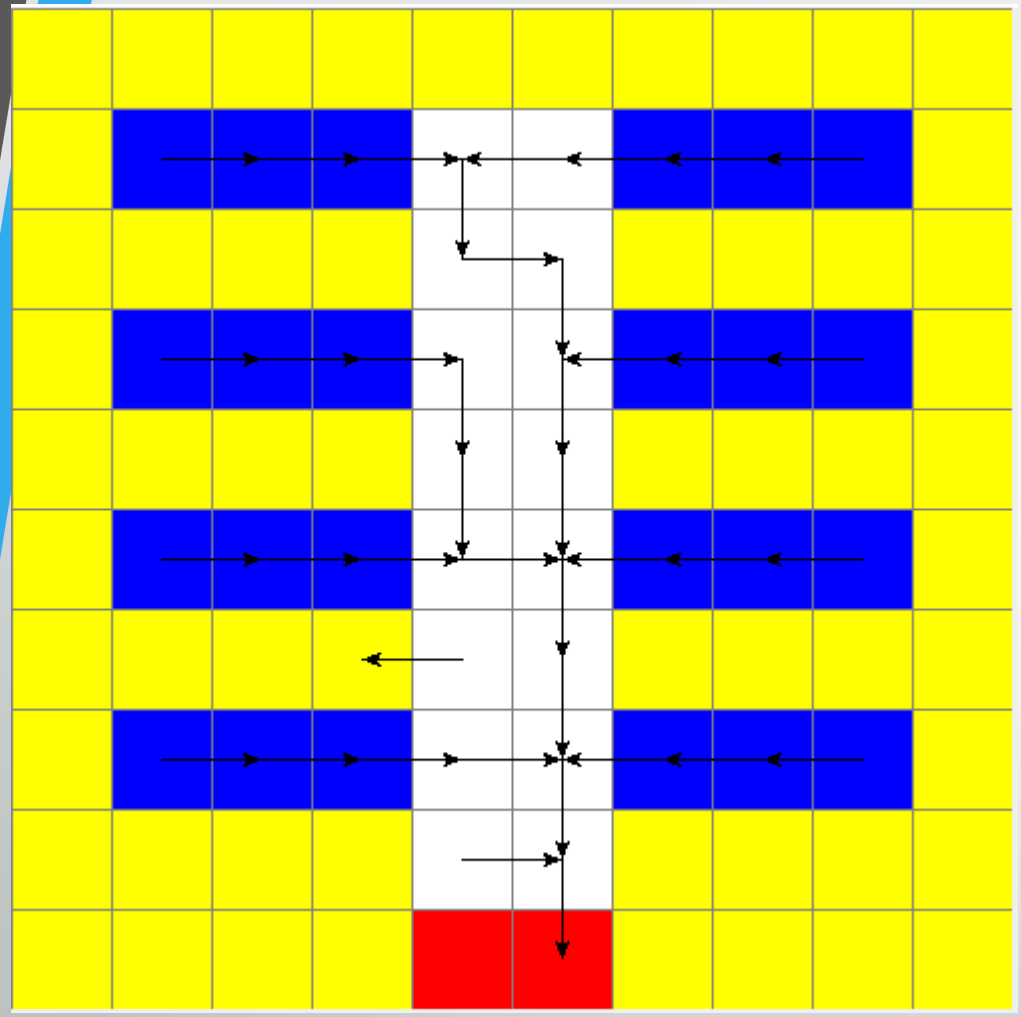
Avec 0 - 250 iterations

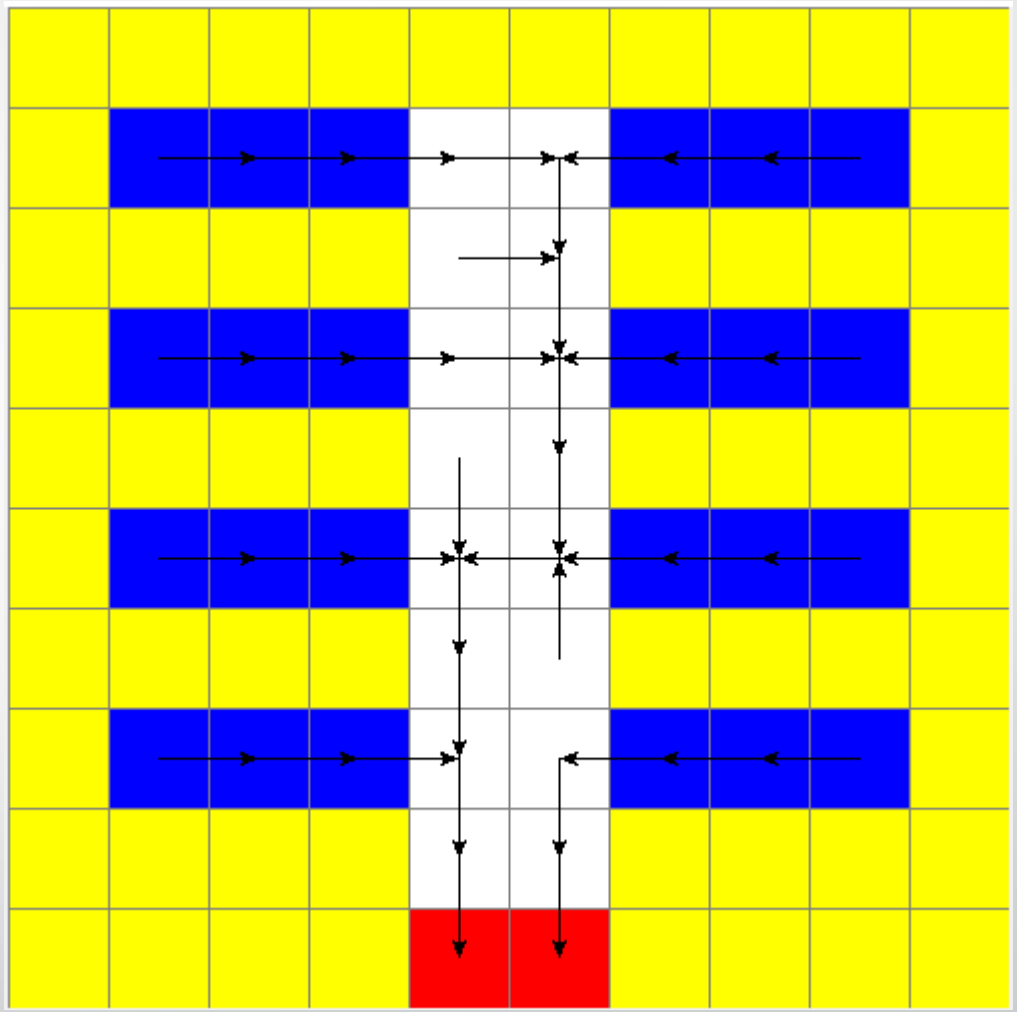


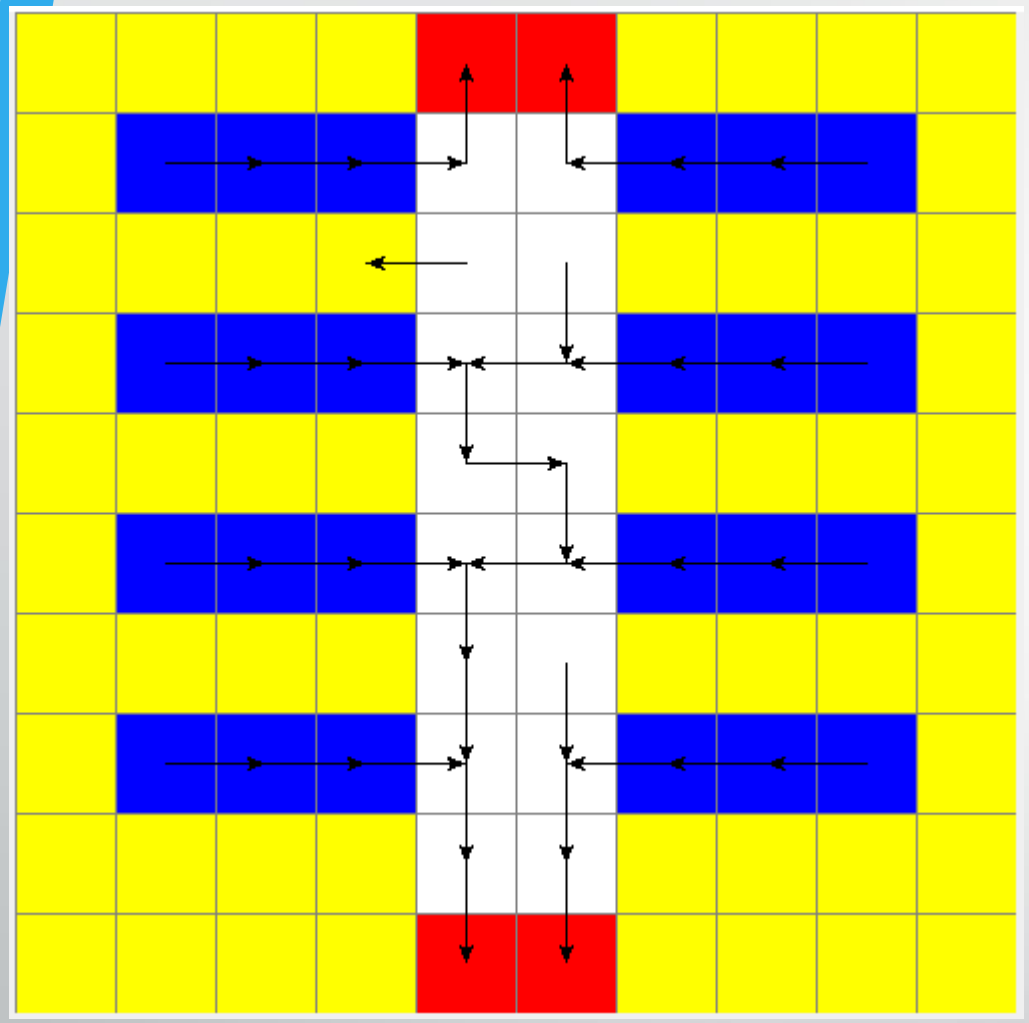


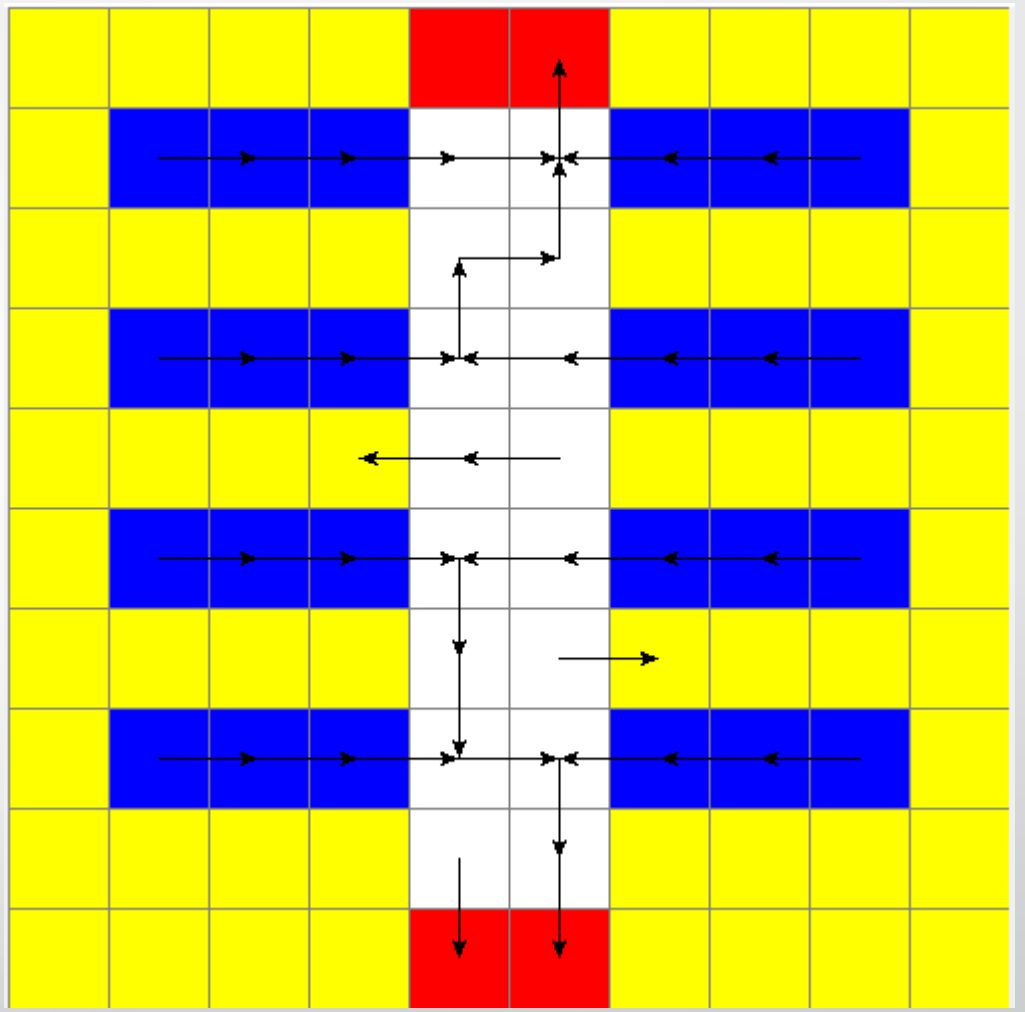












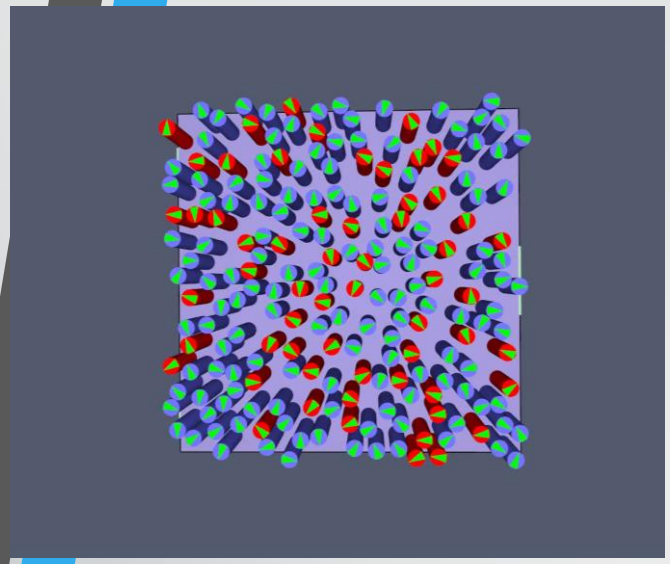
Algorithme genetique

- Réponse au problème
- Sensibilité des paramètres

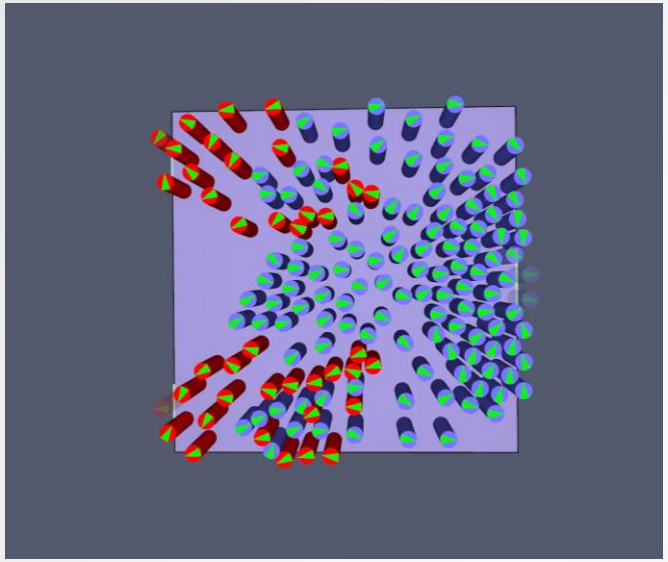
Impact humain

- Complet mais particulièrement difficile à paramétrer

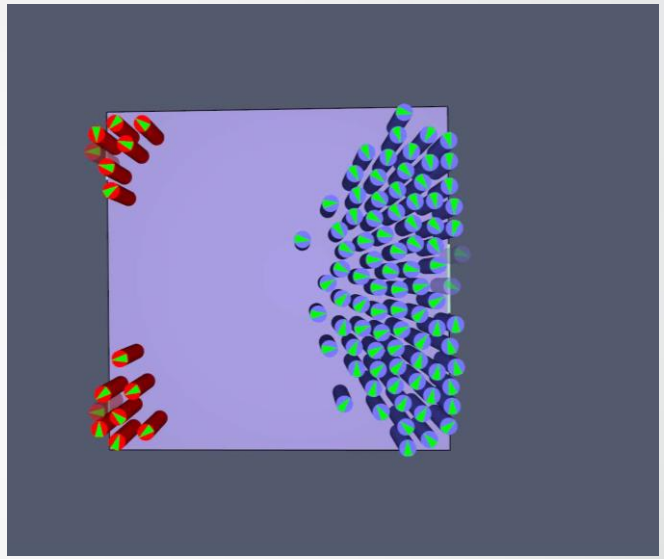
0

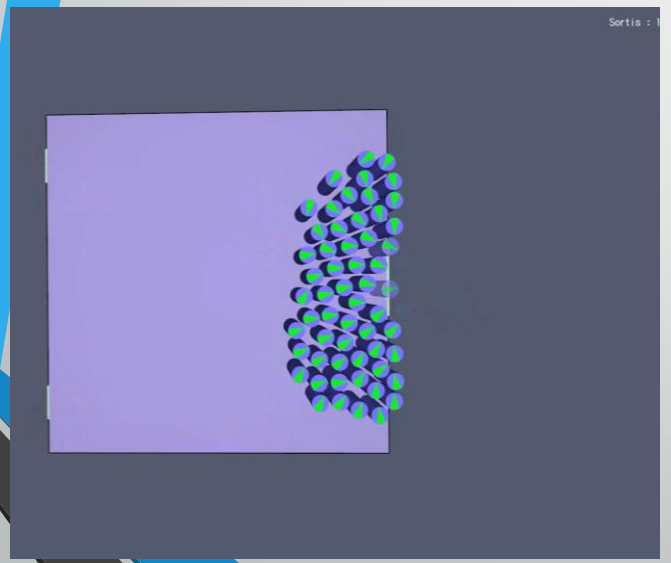


10

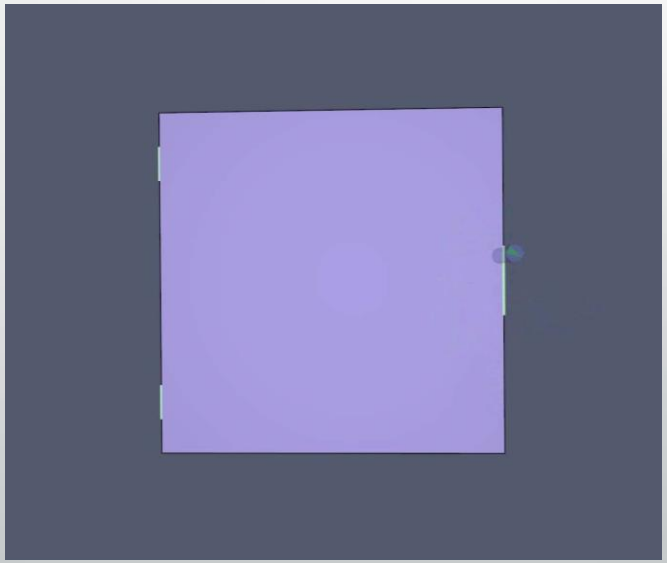


20



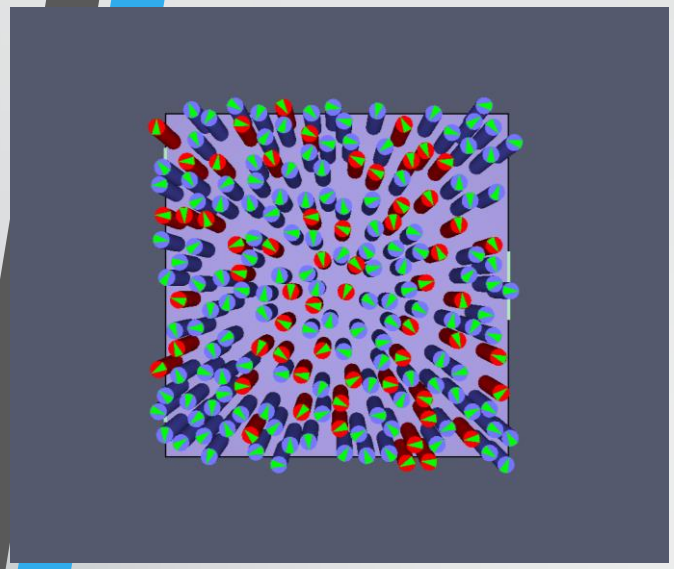


30

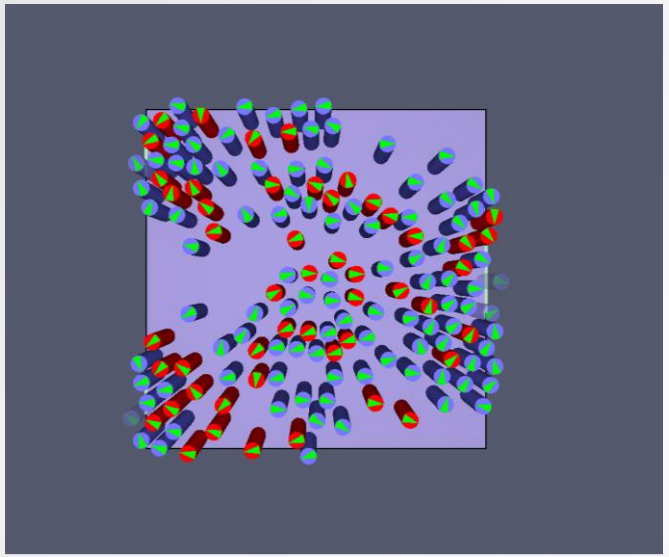


49

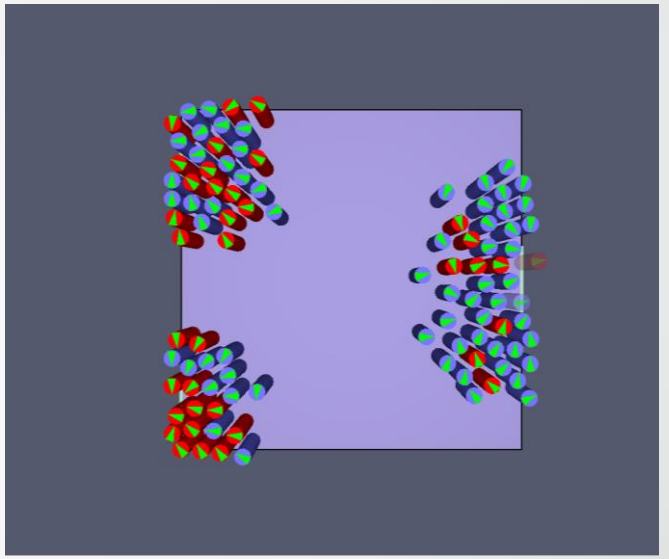
0

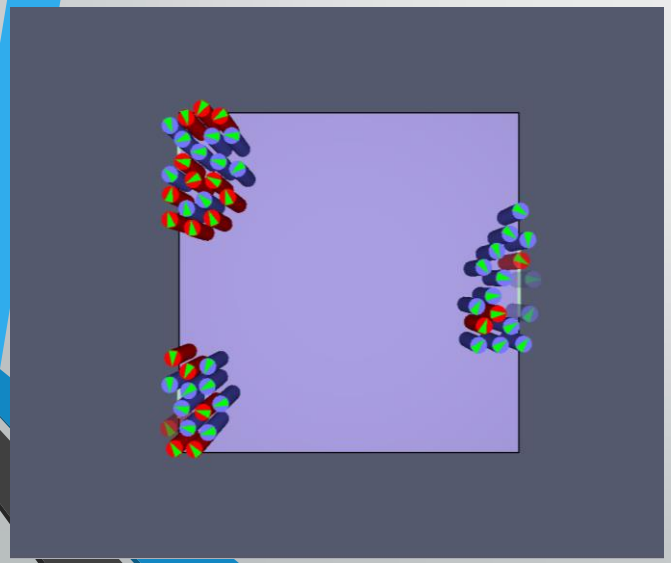


10

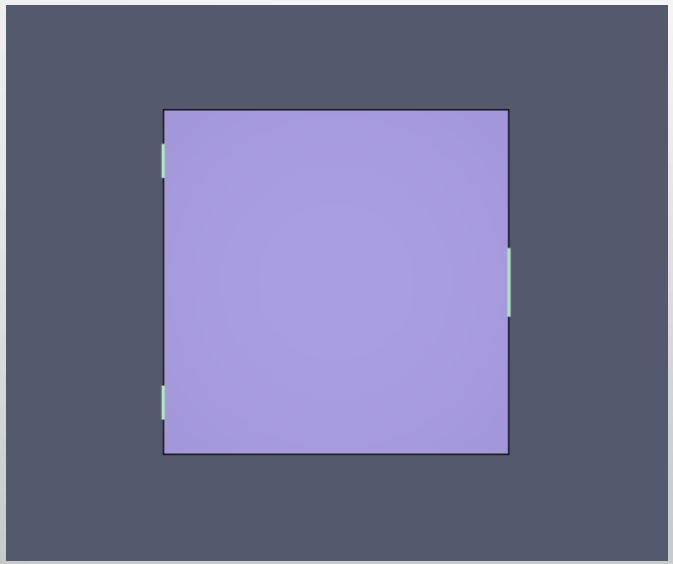


20

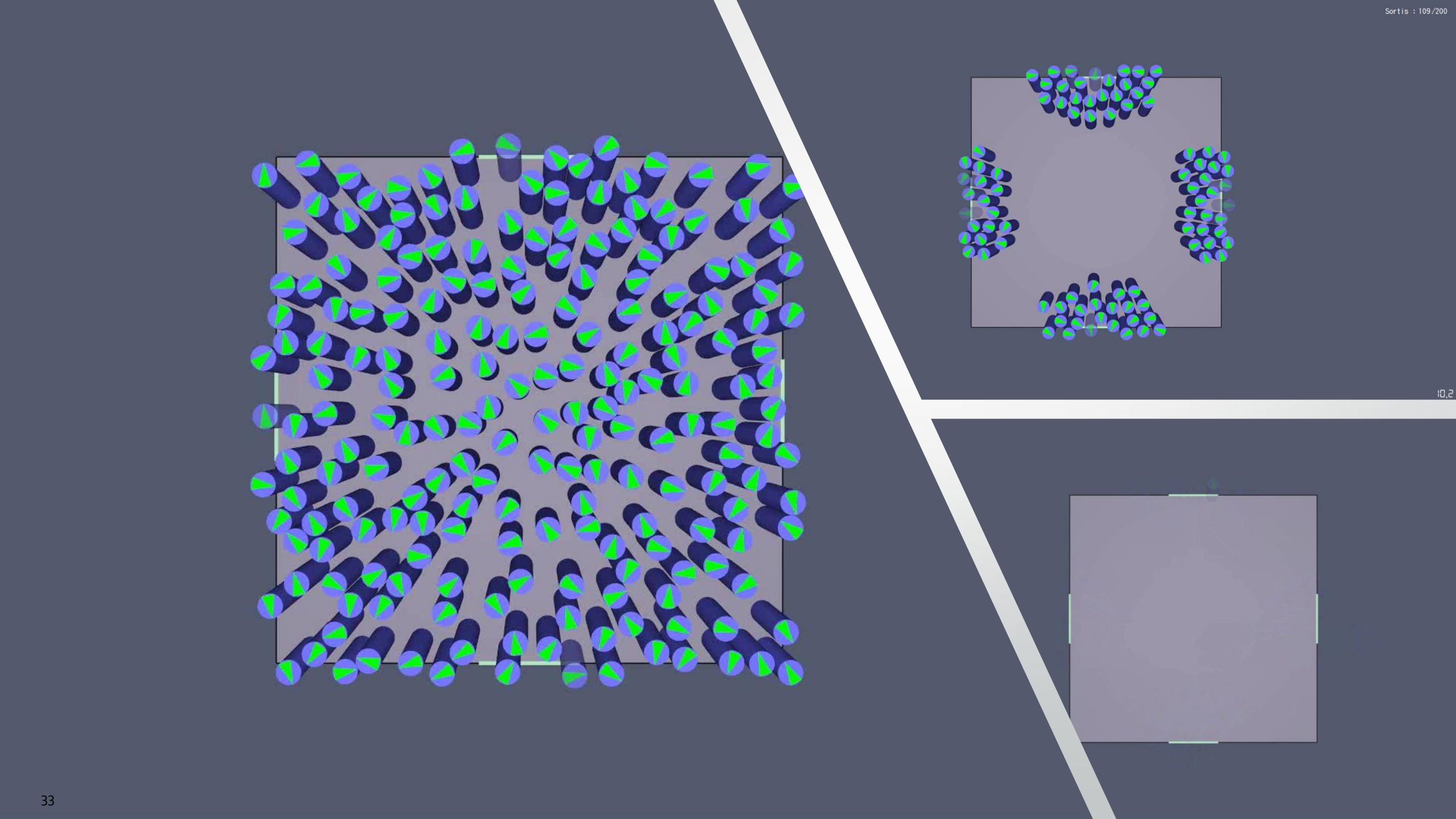


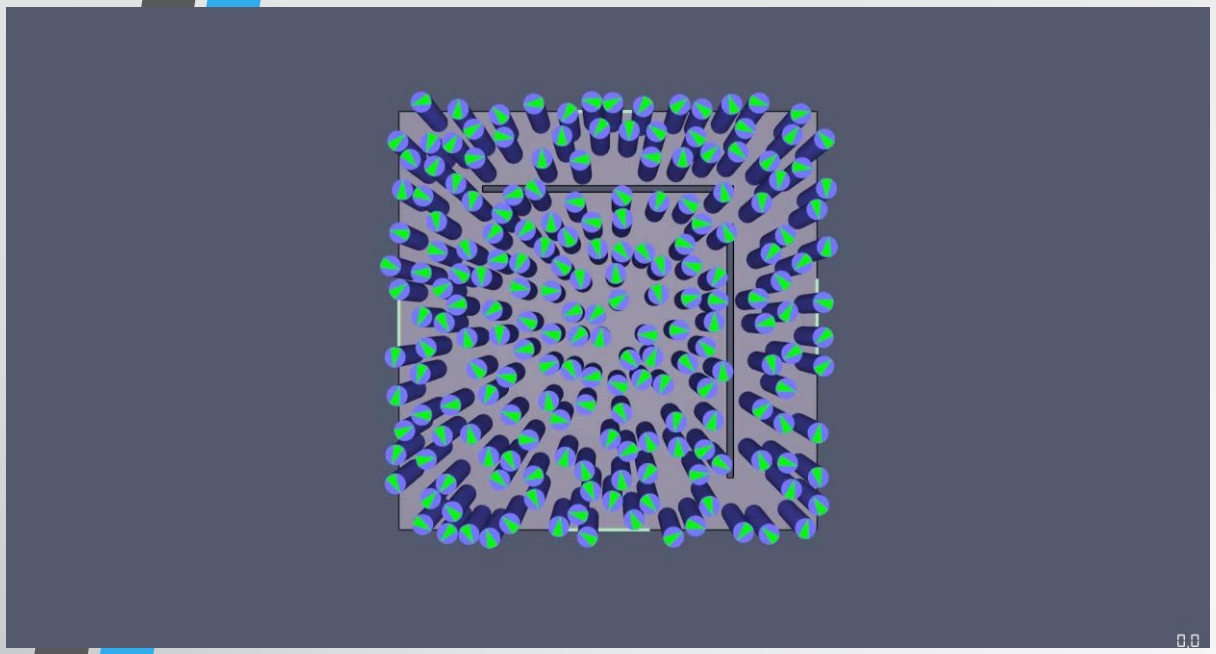


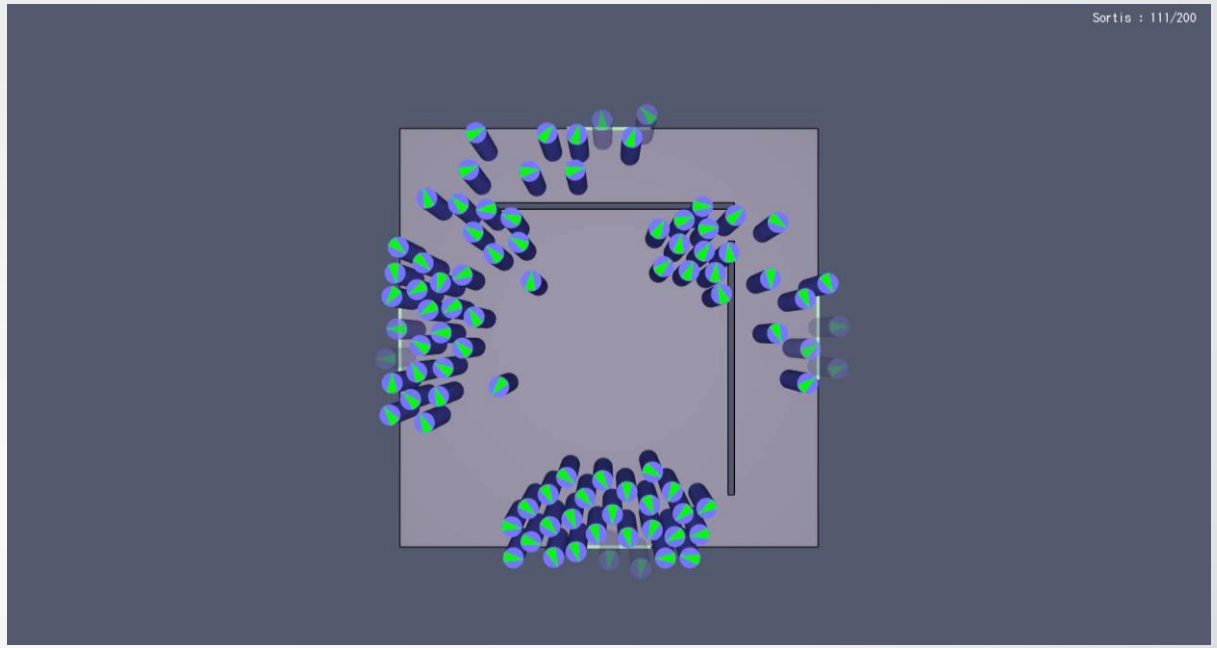
30

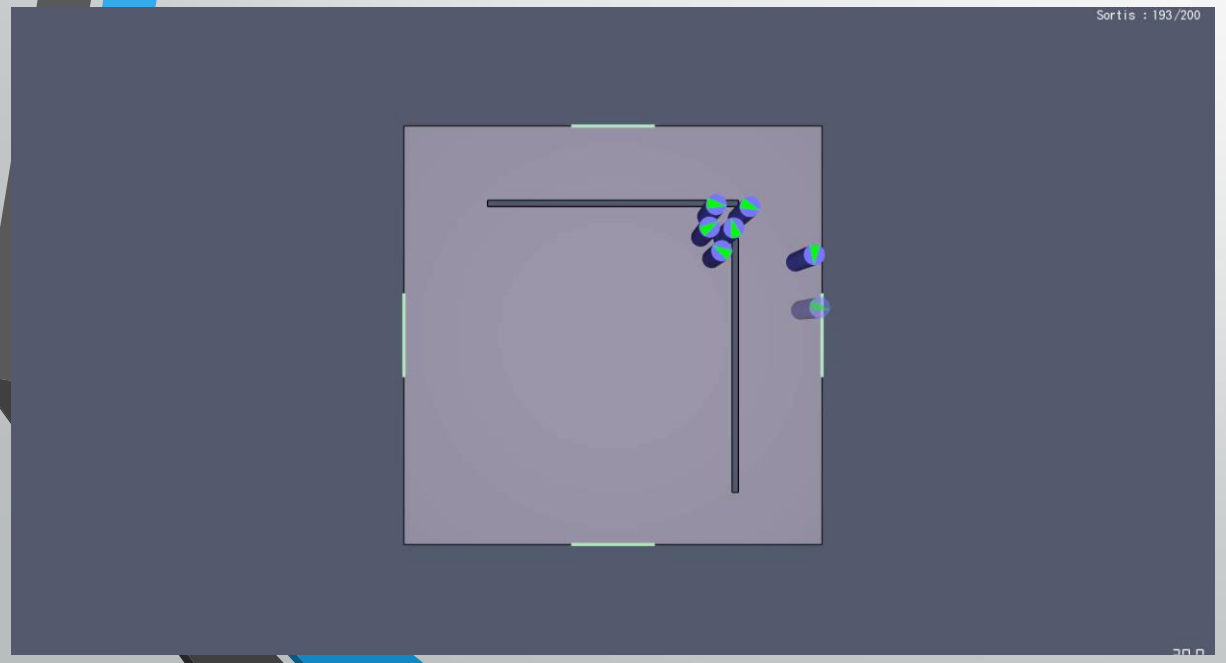


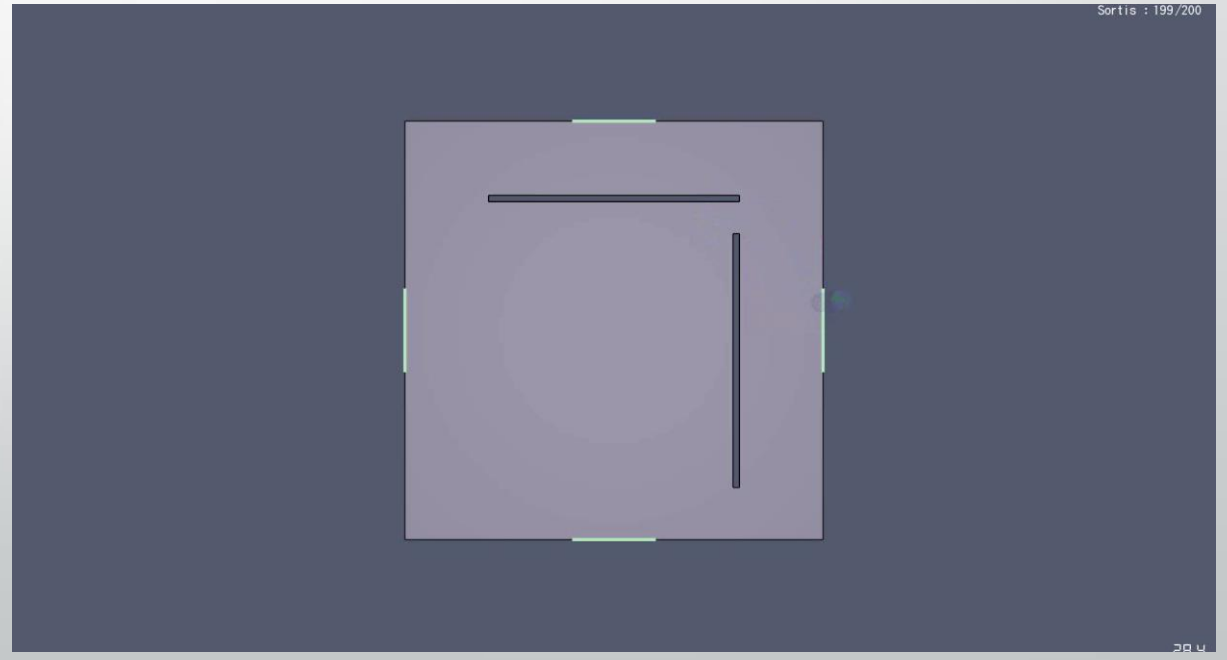
46

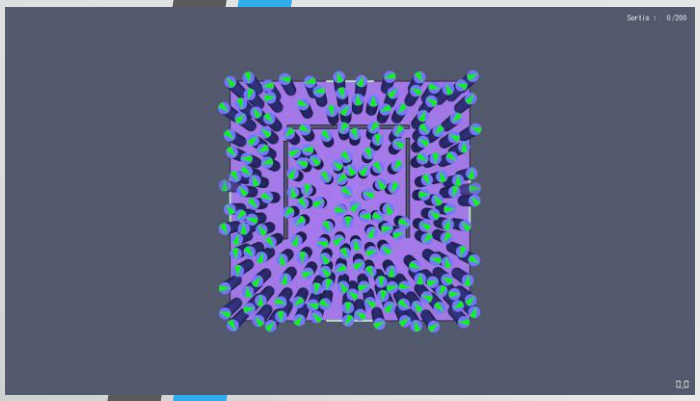

Sortis : 109/200
10,2

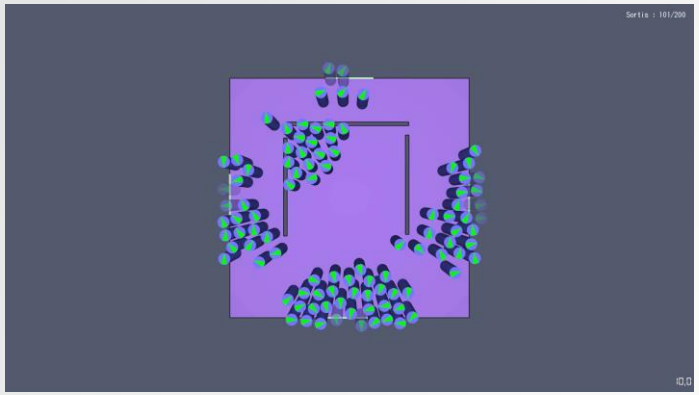

0,0

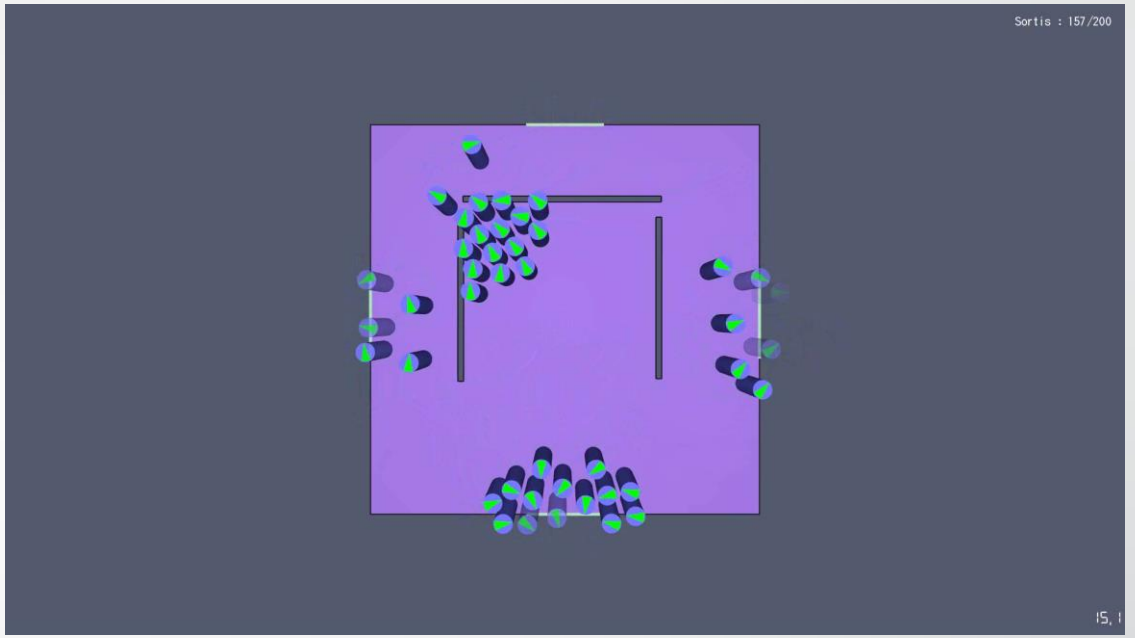

Sortis : 111/200

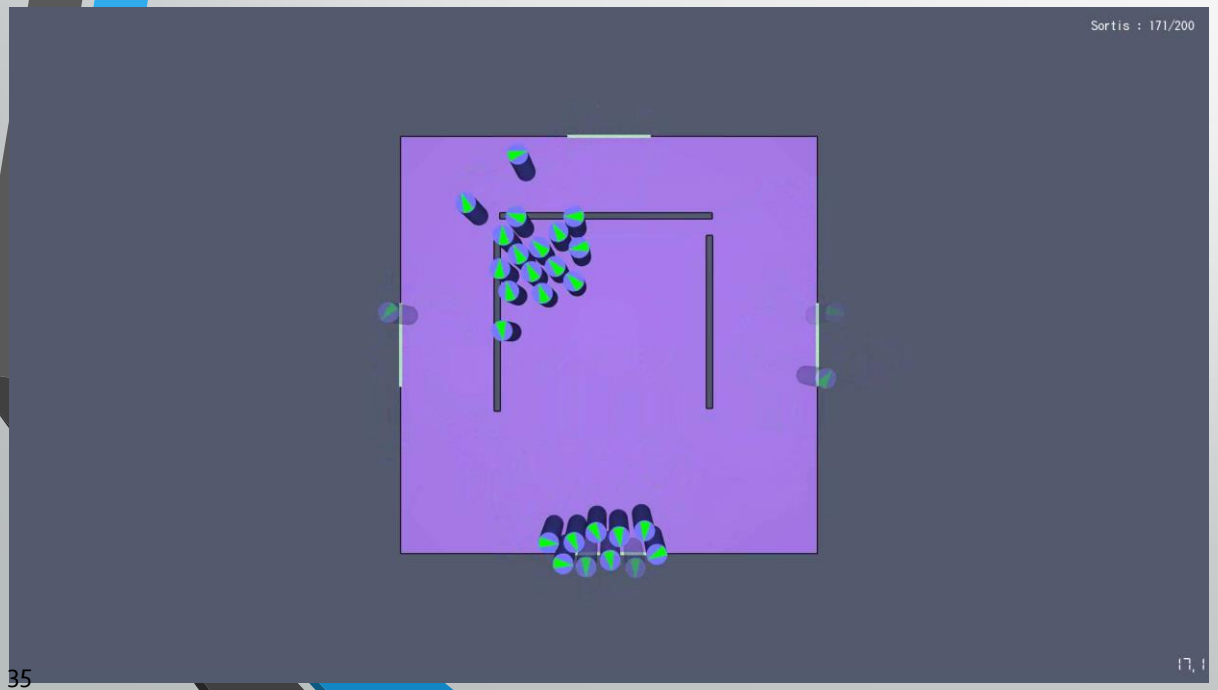

Sortis : 193/200

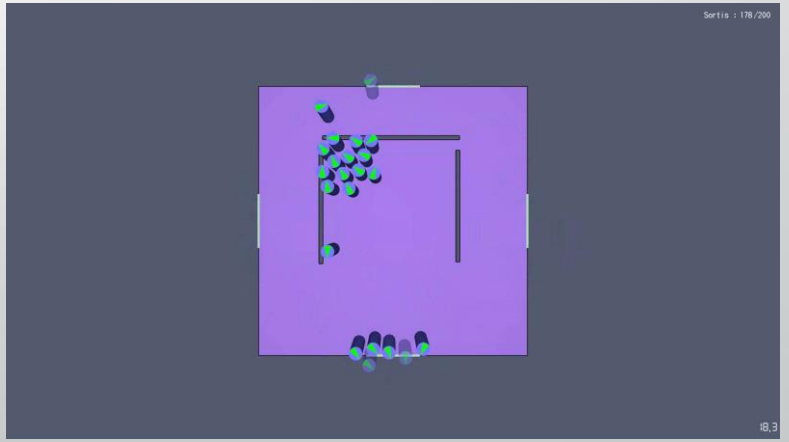

Sortis : 199/200


Sortis : 0/200
0,0


Sortis : 101/200
10,0


Sortis : 157/200
15, 1


Sortis : 171/200
17, 1


Sortis : 178/200
18,3

Conclusion

- Stratégie adaptée à un lieu donné
- La suite :
  - Prise en compte des facteurs humain (changement, suivre, porte par laquelle nous sommes entrés)
  - Stratégie dynamique
  - D'autres méthodes (tels que le gradient)
  - La présence des serres-file, des guides-file et un coordinateur d'évacuation